VLSI Parallel Sorter Architecture for Streaming Data

Dongjae Song, Kyoung Kun Lee, Soongyu Kwon, and Jong Tae Kim
School of Information and Communication Engineering,
Sungkyunkwan University,
Suwon, Gyeonggi-do, South Korea

Abstract - *Data sorting is used indispensably in computer system and affects performance of entire processing. Recently, its importance is increased in many areas from database server system to embedded computer system. Data sorting algorithms and implementation methods have been conducted in many ways that reduces complexity for operation purposes, along with hardware/software structures. In this paper, we propose new sorting architecture that uses 8-way merge sort algorithm to sort streaming data from database. By using pipelined merge sort algorithm, we can organize parallel sorter architecture. Through functional simulation, we analyze relation between amount of data and required clock cycle to sort the data. Also, we can calculate size of memory to implement sorter architecture.*

Keywords: VLSI architecture, hardware sorter, parallel sorter, merge sort

1 Introduction

In most computer system and database system, data sorting is basic and indispensably used operation. Therefore, data sorting takes important part in capability of entire system. Its importance is getting bigger from database server to embedded computer system [4]. Because of its importance, many experiments have been conducted across the board on system before to suggest algorithms that reduces complexity and sorting time for operation purposes, along with hardware/software structures [1][3][5]. Through them, many parts are improved and many algorithms were developed. However, data sorting is still one of the operations that take most time. That is why it is important to see how sorting system, which accompanies sorting application, is constructed. Data sorting algorithms and implementation methods have been conducted in many ways. For example, at algorithmic aspect, quick sort, heap sort or merge sort algorithm can be used. And sorter can consists of hardware only or hardware/software hybrid method [6]. These implementation of the sorter has advantages and disadvantages of each are quite different.

Our objective is to reduce unnecessary time in sorting streaming data that comes through input and develop VLSI architecture of hardware sorter that outputs sorted data. Using 8-way pipelined merge sort algorithm, we design hardware sort unit model using VHDL. Entire sorting system is developed by connecting these sort units. Then, simulate system for verification and performance analysis. Through functional simulations, we derive a simple formula for relationship between amount of data and required clock cycle to sort data.

This paper is comprised as following. Section 2 describes many hardware sorter architecture and pipeline merge sort algorithm. Section 3 describes our approaches to materializing VLSI architecture of hardware sorter. Section 4 shows result of experiment. Section 5 determines conclusion that arises from result of experiment.

2 Related Work

2.1 Pipeline Merge Sort Algorithm

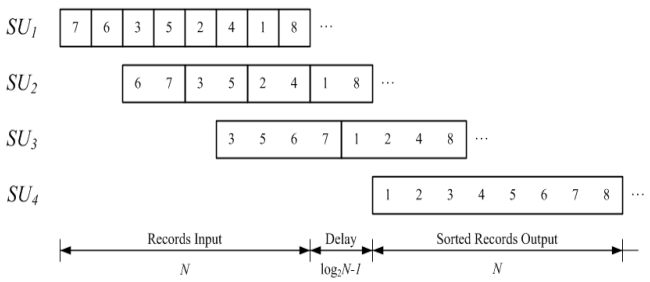

Figure 1 Pipeline merge sort algorithm

Merge sort Algorithm performs multiple levels of merge and compare operation when input is received. Essentially $\log_2 n$ steps are done and applied to the data in 2-way merge sort. This process is done in typical time so each merge step takes O(n) time. Therefore, total operational time costs get O($nlogn$) complexity. In case of 2-way merge sort, the first step is to merge 1 record. Partially sorted records are called strings. The second step merges strings with size of two. m-th step merges string size of 2^{m-1}. Looking at example shown in Figure 1, SU_1(Sort Unit) takes in a series of records and outputs one arranged string from two records. 7 is greater than 6 so 6 comes first and 7 is put at next clock cycle. During the process, next records 3 and 5 are computed. SU_2, which

already has input values of string (6, 7), compares first record of next string 3 with 6 and since 3 is smaller than 6, 3 is output first. Next calculation compares smallest values from remaining records of strings. Through operation, if N records are input, it takes $2N + log_2N-1$ time to finish sorted output. In K-way merge sort, through SU_1, each record is inputted, K records are sequentially ordered to one output string. K strings (K^2 Records), which is the output of SU_1, becomes input of SU_2. To express it in general terms, if one input of SU_m is a string which consists of K^{m-1} records, output becomes a string which has K^m records.

2.2 Hardware Sorter Architecture

Table 1 Comparasion of different sorter architectures

Architecture	Throughput	Input constraint
Parallel	2^l	2^l
Semi-Parallel	2^{l-1}	2^l
Iterative	$< 2^l \times M/((l^2-l+4) \times 2^{l-2}-1)$	$2M$
Pipeline	1	1

Hardware sorter is differentiated in 4 ways depending on how architecture is formed (parallel, semi-parallel, iterative, pipeline). Table 1 shows comparison of throughput and input constraint of different types of sorter. Looking at the throughput, full-parallel show fastest throughput but it takes that much area and gates [7]. M is a degree of parallelism so it lowers hardware efficiency when taking in large values. Therefore, the throughput listed in Table 1 for serial sorters only represents an upper-bound.

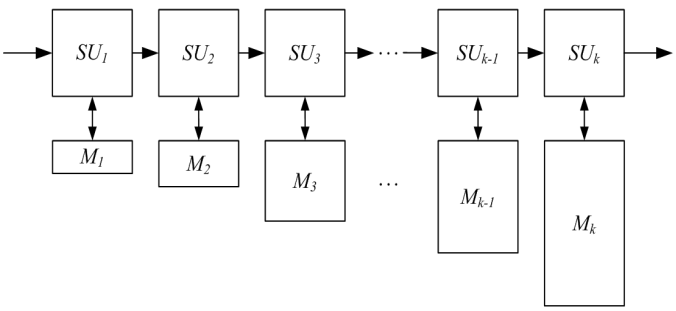

Figure 2 Hardware sorter using external local memories

Hardware sorters in [5] and [6] use local memory which lies outside of needed memory as shown in figure 2. This method uses interface block to control memory. Since this method put memory outside and queue inside of sort unit, complexity of sort unit is reduced. However, additional design is needed for memory interface, which stores date to memory and reuses data.

3 Approach

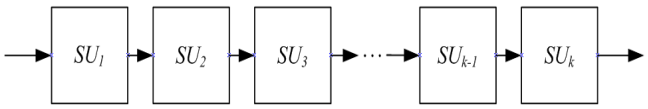

Figure 3 Configuration of suggested hardware sorter

Suggested architecture is materialized by reducing time delay that is addition to time required for merge sort algorithm. Figure 3 is a block diagram that shows structure of hardware sorter. Hardware sorter uses these sort units that are connected in a straight line. In 8-way merge sort, to order 8^k records, it needs k sort units. One sort unit, shown in Figure 4, is comprised of input block, which takes in streaming data from database, disk, or output from previous step, and sort processor block, which compares input data and outputs finished data. Data width is 100-byte (10-byte key) which can send one record in one clock cycle. In our approach, we can extend data sorting capability by connecting sort unit in a straight. Using this method, once clock frequency is determined, we can calculate capability and elapsed time of sorter.

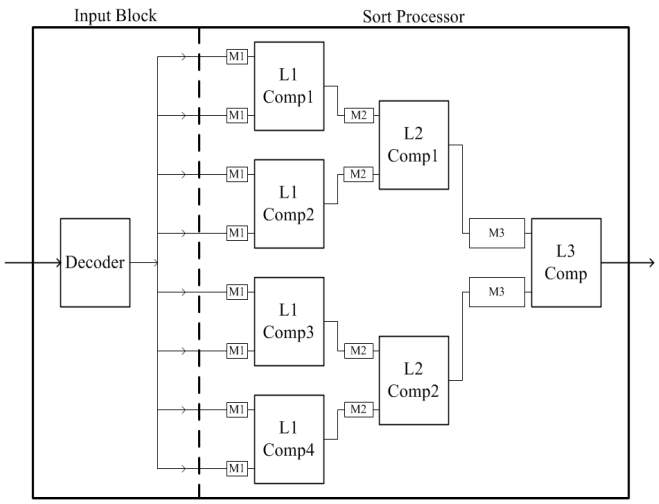

Figure 4 Sort unit block diagram

Our sort processor puts memory inside and seven comparators are comprised in tree structure as shown in Figure 4. Eight inputs of the sort processor are taken in partially ordered strings from input block. Records in input string are stored sequentially in memory. Operation is started when first record from second memory comes in and output comes out after comparison. Input block takes in data of disk or values from previous sort unit to *M1* of sort processor. At this time, according to SU_k, data which is sent *M1* is controlled. For example, in SU_1, one record is sent to each *M1* of sort processor. In case of SU_2, eight records are sent. This

indicates that memory of SU_{k+1} needs K times the capacity of memory of SU_k. To sort one million records, we uses seven sort units through SU_1 to SU_7. To take care of 2^{20} data, 8-way sort is used between SU_1 to SU_6. 4-way sort is used for step SU_7.

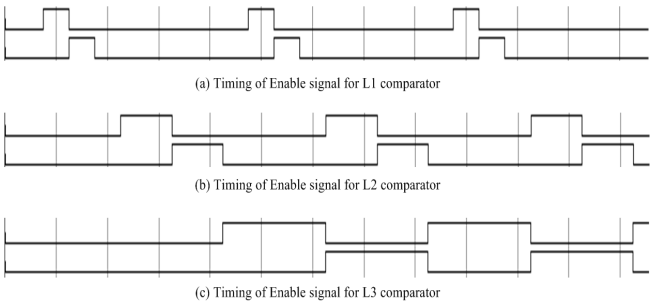

Figure 5 Timing of each comparator enable signal

Each comparator operates while some of input strings are stored. Therefore, comparator needs memory equal to string size. It also needs operational time to finish comparing input string. Figure 5 shows write enable (WE) signal when string input in comparator. Figure 5(a) and (b) are WE signals of *L1* and *L2* comparators. It has no problems since there is plenty of time in clock cycle between first WE signal and next one. Figure 5(c) shows *WE* signal of *L3* comparator that comes in consecutively. In this case, using memory of *L1* and *L2*, two strings can be overwritten.

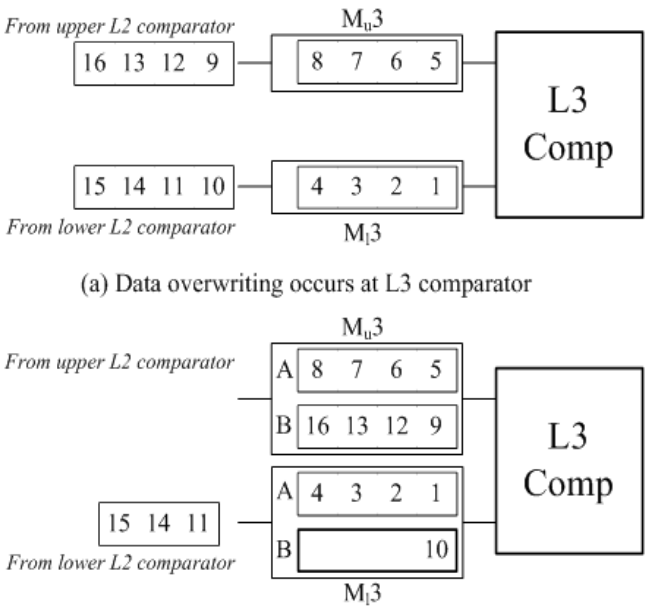

Figure 6 Worst case scenario at L3 comparator

Figure 6(a) shows data overwriting case. If all input records of M_l3(*M3 lower*) is smaller than all of M_u3(*M3 upper)*, next input sorted strings of M_u3 from upper *L2* comparator will overwrite current M_u3 records. To avoid this malfunctional operation, memory size of *L3* comparator is twice as big as size of input string.

4 Simulation Results

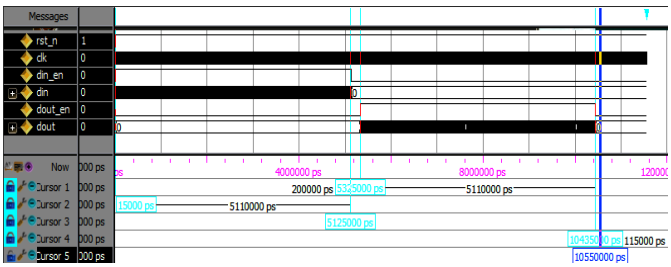

Figure 7 Hardware sorter functional simulation

Using approach of previous section, RTL-code of hardware sorter is developed with VHDL. Figure 7 is result of functional simulation using hardware sorter that was designed with approach of previous section. Sorter that was simulated is designed through SU_1 to SU_3. 512 streaming data was arranged through input. Input was put in 512 clock cycles. After 20 clock cycle wait, 512 clock cycles were needed for output, which confirms pipeline structure with throughout of 1. Simulation clock cycle was 10ns. 512 records were input. After 20 clock cycles, we can see 512 sorted output strings. With design of hardware sorter, one clock cycle delay happens each time it goes through input block and each step of comparator within sort processor. Therefore, when N number of data is input as records, delay that happen when it processed to i level comparator of k-th sort unit is detailed below.

$$\textit{Total Delay Time (clock cycles)}$$
$$= \textit{input records} + \textit{delay} + \textit{output records}$$
$$= N + [\log_2 N + 4(k-1) + i] + N$$

With this relationship, Table 2 is arrangement of sort unit with its input data, delay time, and total delay time. In this table, capability means total records can be sorted by connecting SU_1 to SU_k in a straight line. Delay time means clock cycle difference between end of input records and start of output records. Similar context with delay time, total delay is time takes start of input records to end of output records.

It can be figured out that as input data gets larger, ratio of delay time, excluding time for input and output, gets smaller. Ideally, with 100MHz clock input, data line width being 100-byte, and 100-byte is being input for each clock cycle, a million data can be sorted in 0.02 seconds.

Table 2 Capabilities and delays for each sort unit

Sort Unit	Capability (records)	Delay (cycles)	Total Delay (cycles)
SU_1	8	6	22
$SU_1 \sim SU_2$	64	13	141
$SU_1 \sim SU_3$	512	20	1,044
$SU_1 \sim SU_4$	4,096	27	8,219
$SU_1 \sim SU_5$	32,768	34	65,570
$SU_1 \sim SU_6$	262,144	41	524,329
$SU_1 \sim SU_7$	1,048,576	46	2,097,198

Table 3 Memory size of each sorting unit

(unit : 100-byte)

Sort Unit / Memory	SU_1	SU_2	SU_3	SU_4
M1	1	8	64	512
M2	2	16	128	1024
M3	8	64	512	4096

Sort Unit / Memory	SU_5	SU_6	SU_7
M1	4096	32768	262144
M2	8192	65536	524288
M3	32768	262144	-

Table 3 shows memory size of each sort unit for sorting one million(2^{20}) records. Width of each record is 100-byte include 10-byte key. To sort one million records, 8-way merge sort is used between SU_1 to SU_6 and 4-way merge sort is used for SU_7. Total memory required is about 628.6MB to sort one million records.

5 Conclusion

Sorting is important and necessary part which can affect system performance in the computer system. Its importance is getting larger from database server to embedded computing system. This paper used 8-way merge sort to design architecture of hardware sorter in order to sort streaming data in parallel way. Merge sort is naturally piplelined algorithm. We designed sort processor using merge sort algorithm. Entire system consists of sort units which connected in straight line. This straight line architecture allows each sort unit operate in parallel way. Also, this architecture is scalable for amount of data. Depending size of data to be sorted, we know how to change structure of sorter. Also we can calculate the number of sort units and amount of memory needed. Through simulation, we checked out delay time of suggested hardware sorter is different depending on size of data. We confirmed that delay time is reduced in terms of ratio as more data were sorted, which took advantage of positive trait of pipeline structure.

6 References

[1] A.A. Colavita, et al, "SORTCHIP : A VLSI Implementation of a Hardwar Algorithm for Continuous Data Sorting," *IEEE Journal of Solid-State Circuits*, Vol. 38, pp. 1076-1079, No.6, Jun. 2003.

[2] Anon , et al, "A Measure of Transaction Processing Power," *Datamation*, 31(7):112-118, 1985.

[3] M. Bednara, et al, "Tradeoff Analysis and Architecture Design of a Hybrid Hardware/Software Sorter, " *Proceedings IEEE International Conference on Application-Specific Systems, Architectures, and Processors*, Jul. 2000.

[4] R. Marcelino, et al, "A Comparison of Three Representative Hardware Sorting Units," *IECON '09. 35th Annual Conference of IEEE*, Nov. 2009.

[5] S. Azuma, et al, "DIAPRISM Hardware Sorter," Sort Benchmark, 2000.
http://sortbenchmark.org/Y2000_Datamation_DiaprismSorter.pdf

[6] S. Fushimi, et al. "GREO : A Commercial Database Processor Based on A Pipelined Hardware Sorter," *ACM SIGMOD '93*, vol.22, No.2, pp.449-452, 1993.

[7] Yun-Nan Chang, "Digit-Serial Pipeline Sorter Architecture," Springer, *Journal of Signal Processing Systems*, Vol 61, Issues 2, Nov. 2010.