3A-03

Ultra High-speed FPGA Accelerator for Sorting Application

Ryohei Kobayashi† Kenji Kise†
†Graduate School of Information Science and Engineering
Tokyo Institute of Technology

1 Introduction

In this paper, we propose an approach of sorting acceleration by using a large FPGA. Sorting is an extremely important computation kernel that has been tried to be accelerated in lots of fields. We design and implement the proposed FPGA accelerator, and then evaluate its performance by comparing with a modern desktop computer. From this evaluation, we show how sorting is accelerated.

2 Design and Implementation

Our proposed FPGA accelerator uses two approaches: Sorting Network and Merge Sorter Tree. We describe these methods as follows.

2.1 Sorting Network

A sorting network [1] is an algorithm that sorts a fixed sequence of numbers by using a fixed sequence of comparisons. A sorting network consists of two types of items: comparators and wires. The wires are thought of as running from left to right, carrying values (one per wire) that traverse the network all at the same time. Each comparator connects two wires. When a pair of values, traveling through a pair of wires, encounters a comparator, the comparator swaps the values only if the top wire's value is greater than the bottom wire's value.

Besides, by changing the connection of the comparators, sorting networks can realize lots of sorting algorithms, such as even-odd merge sort, bitonic sort, bubble sort, insertion sort and so on. In [2], sorting networks on FPGAs are discussed in detail.

Figure 1 shows Batcher's odd-even mergesort network with 16-inputs and 16-outputs. Batcher's odd-even mergesort is devised by Ken Batcher for sorting networks of size $O(n\,(\log n)^2)$ and depth $O(\,(\log n)^2)$, where n is the number of items to be sorted. This network is simple, yet efficient and practical. Consequently, we adopt this network as one of the components of our proposed FPGA accelerator.

2.2 Merge Sorter Tree

The merge sorter tree [3] has highly effective performance and good hardware resource usage of the sorting. The merge sorter tree is a data path that executes merge sort and the data path consists of connecting sorter cells as a perfect binary tree. Sorter cells compare two input-values and output one of them depending on its comparison result.

Figure 2 shows how sorting is executed in the merge sorter tree. We define a leaf of the merge sorter tree, as a **way**. For this definition, the merge sorter tree of Figure 2 is 4-way merge sorter tree. We explain how sorting is executed in this 4-way merge sorter tree.

First, at Cycle N, each way outputs integers of "8", "3", "1", "2". Then, "8" and "3", "1" and "2" are compared. Note that the sorted data sequences are stored in the leftmost FIFOs. The sorter cells output smaller elements depending on the comparison result, unless their output FIFOs of the sorter cells are full.

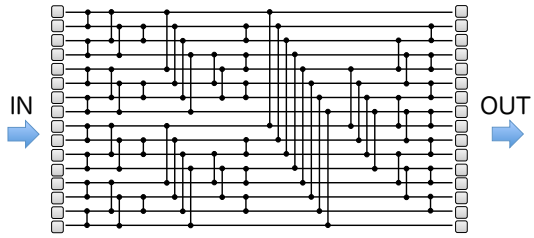

Figure 1: Batcher's Odd-Even Mergesort Network with 16-inputs and 16-outputs

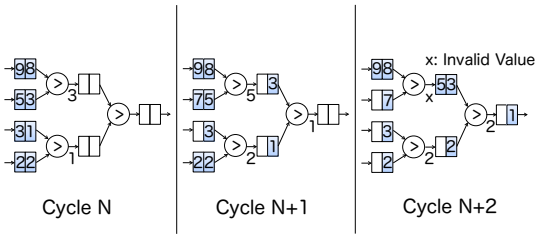

Figure 2: Sorting Process in Merge Sorter Tree

At Cycle N+1, "3" and "1", "8" and "5", "3" and "2" are compared by three sorter cells. All sorter cells output the smaller element of two input elements, because all of the FIFOs are not full.

At Cycle N+2, every sorter cell compares two input elements. The sorter cell that compares "8" and "7" outputs not "7" but an invalid value "x", because its output FIFO having two elements of "5" and "3" is full. If this FIFO is not full, "7" is emitted and is stored in this FIFO. By executing sorting at every sort cell and storing outputs to the FIFO in each cycle, the sorted data sequence is emitted from the root of the merge sorter tree.

2.3 Implemented Sorting Logic

As a platform for the proposed FPGA accelerator, we use the Virtex-7 FPGA VC707 evaluation kit. This kit originally has the Virtex-7 XC7VX485T and 1GB DDR3 SO-DIMM memory, however, we replace this memory with 4GB DDR3 SO-DIMM memory in order to sort larger data sequences.

Figure 3 shows a data path of the sorting logic implemented on the FPGA. The dotted line region in Figure 3 is the merge sorter tree, which executes the main part of sorting. Squares and circles in the merge sorter tree represent FIFOs and sorter cells respectively. Initial Data Generator is used to generate an initial data se-

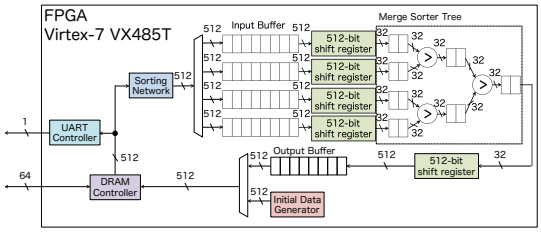

Figure 3: Data Path of Sorting Logic Implemented on the FPGA

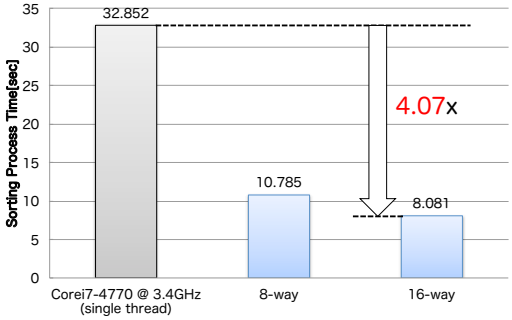

Figure 4: The Sorting Process Time of the Proposed FPGA Accelerator and Single Thread of Intel Corei7-4770 operating at 3.4GHz

quence sorted by the FPGA accelerator. At first, the initial data sequence is stored in DRAM via DRAM Controller. After that, this data sequence is read from DRAM, and then is carried to Sorting Network. This Sorting Network sorts sixteen elements each (one element data size is 32-bit, hence, 512-bit input data consists of sixteen elements) of the data sequence, and then these elements are stored in Input Buffers. Input Buffer in Figure 3 is a FIFO, emitted data from which is carried, one element by one element, to the merge sorter tree by 512-bit shift register. Emitted elements from the merge sorter tree are packed into a 512-bit data by 512-bit shift register, and then are stored in Output Buffer. Output Buffer is also a FIFO, emitted data from which is stored in DRAM. After the data sequence is fully sorted, for verification, this sorted data sequence is displayed on a serial console via UART Controller.

We implemented the sorting logic (shown in Figure 3) in Verilog HDL. To implement DRAM Controller, we use an IP core provided by Xilinx [4]. As a synthesis tool, we use Xilinx ISE 14.7 (Optimization Goal: Speed, Optimization Effort: High). All implemented logics on the FPGA operate at 200MHz and DRAM operates at 800MHz. Consequently, the maximum data-transfer speed is 12.8GB/s between the FPGA and the DRAM.

3 Evaluation

We evaluate the sorting effective performance of the FPGA accelerator when the number of ways is 8 and 16, compared with Intel Corei7-4770 operating at 3.4GHz.

We adopt merge sort as an algorithm running on Intel Corei7-4770. Generally speaking, quick sort is known to be the fastest sorting algorithm, however, quick sort depends on the input-data sequence type. For example, if input-data sequence type is sorted data sequence in reverse order, the computational complexity of quick sort is the worst-case complexity $O(n^2)$. For this aspect, we figure out merge sort is more practical sorting algorithm than quick sort because merge sort is insulated from the influence of the input-data sequence type. We code merge sort in C, compile it with gcc 4.8.2 (-O3 optimization), and measure the sorting process time. Merge sort is executed as single thread of Intel Corei7-4770.

Figure 4 shows the sorting process time of the proposed FPGA accelerator and single thread of Intel Corei7-4770 operating at 3.4GHz, when the number of sorted elements is 256M, whose data type is integer (4 bytes). To measure the sorting process time of the FPGA accelerator, the number of cycles, which the FPGA accelerator takes for sorting, is stored in a register and is displayed on a serial console via UART Controller after sorting is terminated. The sorting process time of the FPGA accelerator with 8-way merge sorter tree and 16-way merge sorter is 10.785 sec and 8.081 sec respectively, while that of single thread of Intel Corei7-4770 operating 3.4GHz is 32.852 sec. In other words, the proposed FPGA accelerator can obtain higher performance of up to 4.07x than Intel Corei7-4770 operating at 3.4GHz.

4 Conclusion

In this paper, we proposed an approach of sorting acceleration by using a large FPGA, described design and implementation, and evaluated the sorting effective performance, compared with Intel Corei7-4770 operating at 3.4GHz. As a result, the proposed FPGA accelerator can obtain higher performance of up to 4.07x than Intel Corei7-4770 operating at 3.4GHz.

There is still room for improvement of the FPGA accelerator, because hardware resource usage is about one-fifth, even if the number of ways is 16. Therefore, our future works are to expand features, such as the number of ways, and to evaluate its sorting effective performance.

References

[1] Donald E. Knuth. *The Art of Computer Programming, Volume 3: (2Nd Ed.) Sorting and Searching*. Addison Wesley Longman Publishing Co., Inc., Redwood City, CA, USA, 1998.

[2] Rene Mueller, Jens Teubner, and Gustavo Alonso. Sorting networks on fpgas. *The VLDB Journal*, Vol. 21, No. 1, pp. 1–23, February 2012.

[3] Dirk Koch and Jim Torresen. Fpgasort: A high performance sorting architecture exploiting run-time reconfiguration on fpgas for large problem sorting. In *Proceedings of the 19th ACM/SIGDA International Symposium on Field Programmable Gate Arrays*, FPGA '11, pp. 45–54, New York, NY, USA, 2011. ACM.

[4] Memory Interface Generator (MIG). http://www.xilinx.com/products/intellectual-property/MIG.htm.