Area and Throughput Aware Comparator Networks Optimization for Parallel Data Processing on FPGA

Christophe Layer, Daniel Schaupp, Hans-Jörg Pfleiderer
University of Ulm, Institute of Microelectronics
Albert-Einstein-Allee 43, 89081 Ulm, Germany
Email: christophe.layer@uni-ulm.de

***Abstract*— As hardware synthesis tries to go behavioral, system designers tend to neglect low level optimizations, e.g., retiming or resources reuse. However, complex architectures such as highly parallel sorter cores cannot be directly packed into standard devices by the Electronic Design Automation (EDA) tools and often needs fundamental rethinking. Therefore this paper resumes a cost study of the hardware realization of a recurrent parameterizable N-sorter based on Batcher's bitonic algorithm and shows how to recombine different network sizes by preserving data throughput. With the time complexity of the original bitonic sorter, the recurrent architecture yields a lower area complexity by reducing the communication within the network and minimizes the cost in terms of hardware resources. Furthermore, the validity domain of internal architectural parameter combinations demonstrate the existence of an optimal area-throughput trade-off for Very Large Scale Integration (VLSI) models based on Field Programmable Gate Array (FPGA) synthesis, as well as a full scalability of the highly parallel sorting scheme.**

I. Introduction

Following Moore's law, the number of gates and features inside FPGAs has increased so dramatically that they have progressed to a point where System on Chip (SoC) designs can be built on a single device. They compete with capabilities that have traditionally been supplied through Application Specific Integrated Circuit (ASIC) devices only and offer a lot of advantages compared to Digital Signal Processors (DSPs) or general purpose processors when it comes to SoC design. Not only do they give system designers the possibility to parallelize their architectures, but they also provide an excellent density of hardware Intellectual Property (IP) and lower the price per Giga Operations Per Second (GOPS) rapidly. Moreover, it has been shown that they achieve greater performance per unit of silicon area than processors [5]. But as today's EDA products can map software designs into a hardware description language implementation suitable for FPGA synthesis, this paper shows the importance of reconsidering design restructuration and retiming, via the hardware implementation of a nontrivial architectural paradigm, namely parallel data sorting.

Hence after reminding first the fundamentals of Batcher's bitonic sorting algorithm and its realization at the register transfer level, this paper presents optimization methodologies for the engineering of a sorter core based on Stone's perfect shuffle network targeting area reduction. However, since area-time trade-offs always have to be made, we discuss the results of the hardware implementation of different networking schemes on various FPGA devices by means of a relative cost function.

II. Bitonic Merging Networks

A. Parallel Architecture

Batcher's bitonic merger and odd-even transposition sorter [1] are based on a comparison network scheme in which many compare and exchange operations are performed in parallel. A particularity of this network is that the sequence of comparisons is absolutely deterministic and depends neither upon the initial state of the file of keys to be sorted, nor upon the result of the previous comparison step. Fig. 1 represents Batcher's bitonic sorting network in a Knuth diagram for an input of $N = 16$ keys, using both ascending "↑" and descending "↓" comparators. According to text's conventions for drawing networks [8], an arrow from x_i to x_j is used to indicate that the larger number goes to the point of the arrow and the smaller to the base. This network consists of $n = \log_2 N$ successive merging phases i with $1 \leq i \leq n$, in which pairs of sorted sequences of length 2^{i-1} are presented in oppositely sorted order and then merged together, according to the bitonic principle [1]. A naive layout of this network would involve bringing into adjacent lines the pairs of inputs keys to be compared and then returning them to their original positions in the appropriate order. An obvious improvement would be not to return the required inputs keys after bringing them together but remember their logical positions. As many research groups have enhanced the bitonic architecture [6], [15], we chose one which allows further modifications of both the hardware size and the internal scheduling.

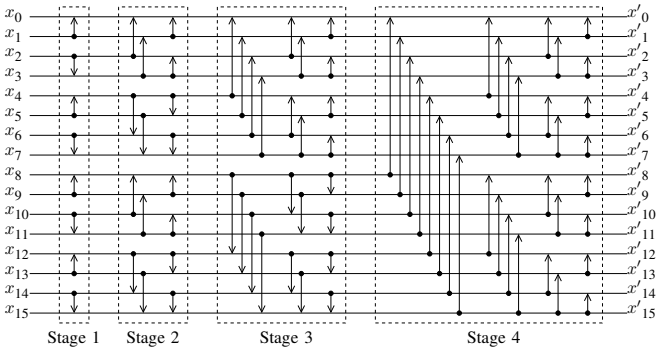

Fig. 1. Batcher's bitonic sorting network represented in a Knuth diagram for an input of $N = 16$ keys has an $O(\log^2 N)$ time complexity. Each of the N keys is L bits in length.

B. Recurrent Architecture

Stone [13] showed that a sorting network for $N = 2^n$ elements could be constructed by following a regular pattern on which each of the n steps consists of a perfect shuffle of the first $2^{n-1} = \frac{N}{2}$ elements with the last $\frac{N}{2}$, followed by simultaneous operations performed on $\frac{N}{2}$ pairs of adjacent elements [8]. The resulting architecture depicted in Fig. 2 includes a regular pattern of ascending comparisons "↑", descending comparisons "↓" and neutral operations "∅". During a stage s, for $s < n$, $n-s$ steps perform "∅" operations at first. They are succeeded by s steps in which the operations within step t consist alternately of 2^{t-1} "↑" followed by "↓". During the last stage, all operations are "↑" and constitute in fact an N keys bitonic merger for two monotonic series of length $\frac{N}{2}$, as considered in Section III.

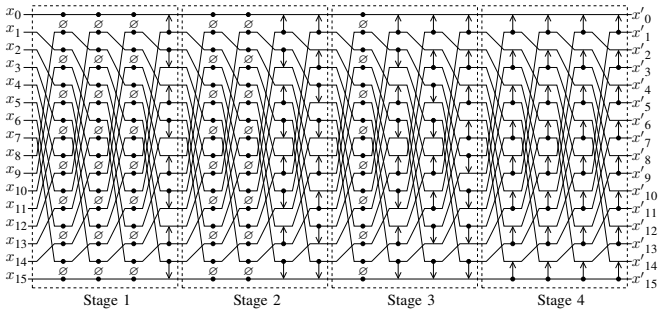

Fig. 2. Regular structure of the bitonic sorting network using Stone's perfect shuffle on $\log_2 N$ stages with $\log_2 N$ steps per stage.

Using recirculation [10] and a set of N registers, this network structure can be simplified to a single column of $N/2$ special controlled switching comparators noted later "$\updownarrow$" as in Fig. 5, executing a predefined sequence of operations, according to [9]. The final remaining time for this model is then expressed through

$$t_{\mathrm{s}}(N) = \left((\log_2 N)^2 - \log_2 N + 1\right) \cdot T_{\mathrm{clk}}, \tag{1}$$

where $f_{\mathrm{clk}} = T_{\mathrm{clk}}^{-1}$ represents the frequency at which the recurrent architecture can be clocked. In this sorting scheme, the keys are input at once and processed in parallel. Their length L does not influence $t_{\mathrm{s}}(N)$. While the time complexity $O(\log^2 N)$ of the sorter is the same as the original bitonic sorting network, its complexity in terms of logic gates and comparators has been drastically reduced from $O(N \log^2 N)$ to $O(N)$. However, for large N and L, as it is the case in most applications, the practical realization of such a network yields a moderate performance due to the high routing density of the sorter. Therefore, we have to consider further optimization methods for an efficient implementation of the bitonic merger into physical devices, e.g., FPGAs.

III. Optimization Methodologies

A. Network Retiming

When it comes to the hardware implementation of such a sorting network that includes a perfect-shuffle mapping, a price has to be paid in terms of routing resources, especially in FPGAs. Previous studies [4], [6], [11], [16] have shown the influence of the butterfly pattern over the complete architecture of the system. The recurrent bitonic sorting network presented in the previous section is scalable in the sense that it is adaptable to any kind of bus width, as long as the number of keys to sort is a power of two ($N = 2^n$). Slightly diverging realizations of redistributing networks for sorting were found in the literature where the main idea was either to use special design environments to verify the lengths of intermediate wires [2] or to resort to the parity strategy to reduce the communication within the sorter [10].

The introduction of a new parameter p representing the degree of parallelism inside the network means a trade-off between area and throughput for the sorter. It corresponds to the granularity at which the keys are internally processed in the comparators and within the network. In this sense, with L the length of one key, if $p = 1$ corresponds to the completely serial solution and $p = L$ the full parallel solution, then (1) becomes

$$t_{\mathrm{s}}(N,p) = \left((\log_2 N)^2 - \log_2 N + 1\right) \cdot \left\lceil \frac{L}{p} \right\rceil \cdot T_{\mathrm{clk}}, \tag{2}$$

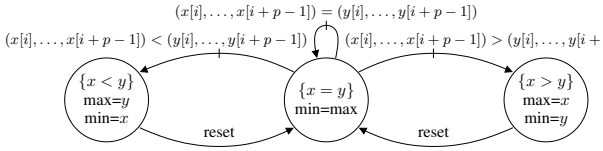

Fig. 3. Deterministic finite state automaton used during a partial comparison for the retiming of the comparator modules with an internal degree of parallelism p in the pseudo-serial processing.

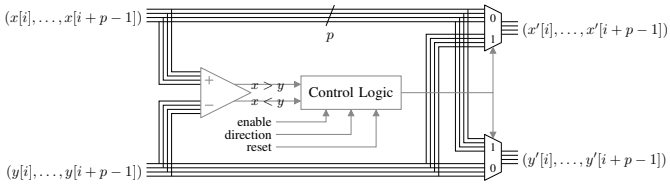

Fig. 4. Internal representation of a controlled switching comparator which processes $p = 4$ bits in parallel. The control logic includes the state machine depicted in Fig. 3 and decides to output the keys x and y on x' or y'.

so that the recirculation of one key through one level of comparators requires $\lceil (L/p) \rceil$ clock cycles. The main benefit is for the area A of the sorter which varies in $O\left(N^2 p^2\right)$ [6], [11]. Though the throughput is accordingly reduced, it is foreseeable that the global routing delay might at our advantage be shortened as well.

Fig. 3 shows the state diagram for a pseudo serial comparison element in which the keys are inserted MSBs (Most Significant Bits) first, p bits at the time. In the initial state, two keys x and y are considered to be equal. This remains as long as all the processed p bits don't differ respectively in x and y. Otherwise, one of the two other states gets selected and holds until the end of the processing of the L bits. Fig. 4 shows the internal structure of a controlled switching comparator as used within a recurrent network architecture according to [9], where the L bit keys are processed p bits at the time.

B. Network Recombination

Per definition, a bitonic sorter of order N is a comparator network capable of sorting any bitonic sequence of length N into nondecreasing order, whereas a sequence $\langle x_1, \ldots, x_N \rangle$ is bitonic if $x_1 \geq \cdots \geq x_k \leq \cdots \leq x_N$ for some k, $1 \leq k \leq N$ [8]. The problem of merging $x_1 \leq \cdots \leq x_N$ with $y_1 \leq \cdots \leq y_N$ is a special case of the sorting problem, since merging can be done by applying a bitonic sorter of order $2N$ to the sequence $\langle x_N, \ldots, x_1, y_1, \ldots, y_N \rangle$.

Although a decomposition of the standard bitonic sorter works with arbitrary or mixed sizes [12], we chose to implement recursively an N keys sorting engine based on one level of $M = 2^m$ bitonic sorters for $\frac{N}{M}$ keys and m levels of bitonic mergers for each time twice the previous amount of keys, all with recirculation. As seen in Fig. 5, the two kinds of networks, i.e., sorting and then merging, can handle $(m+1)N$ keys in parallel, pipelined over $m+1$ levels. The recirculation channels include a direct feedback word-wise of N times p bits to each key. Compared to $t_{\mathrm{s}}(N,p)$ for sorting N keys, the time $t_{\mathrm{m}}(N,p)$ needed for merging two sets of $\frac{N}{2}$ keys is much shorter and yields

$$t_{\mathrm{m}}(N,p) = (\log_2 N) \cdot \left\lceil \frac{L}{p} \right\rceil \cdot T_{\mathrm{clk}}. \tag{3}$$

For hardware simplification reasons, the design includes only one single clock domain. Between sorters and merger, the remapping $p_1 \mapsto p_2$ is simply done using shift registers, depicted in grey in Fig. 5. If necessary, the serial insertion of zero-bits can be used

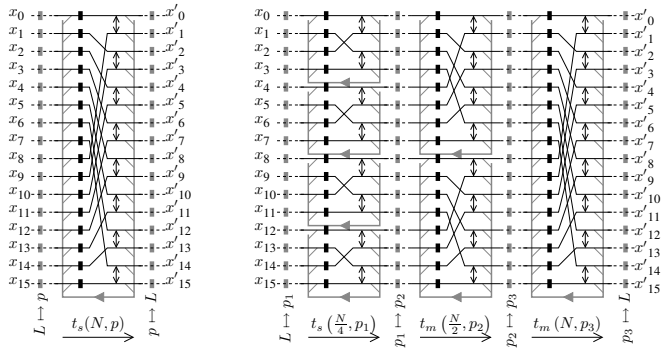

Fig. 5. Recombination of recirculating bitonic merging networks using one $M = 1$ (left) or three feedback levels $M = 4$ (right) to sort $N = 16$ keys.

to resynchronize a bitonic network without influencing the sorting functionality. The condition for achieving the maximum throughput is that the time $t_\mathrm{s}(\frac{N}{M}, p_1)$ in $O\left(\log^2 \frac{N}{M}\right)$ needed to sort $\frac{N}{M}$ keys with a granularity p_1 is at least equal to the time $t_\mathrm{m}(N, p_i)$ in $O(\log N)$ needed to merge the resulting keys, i.e., never becoming a bottleneck, such that

$$t_\mathrm{s}\left(\frac{N}{M}, p_1\right) \geq \max\left\{t_\mathrm{m}\left(\frac{N \cdot 2^i}{2M}, p_i\right)\right\} \quad \forall 2 \leq i \leq m+1. \quad (4)$$

Hence we obtain a relationship between p_1 and p_i as we try to minimize the overall area of the architecture considering the worst case with $i = m + 1$

$$p_1 \leq \frac{p_i}{\log_2 N} \cdot \left(\left(\log_2 \frac{N}{M}\right)^2 - \log_2 \frac{N}{M} + 1\right), \quad (5)$$

whereas since both are integers, p_i might be set to the next suitable value to ensure the highest throughput.

IV. Hardware Implementation Results

A. Area and Throughput

In order to be able to evaluate the complete sorting unit, we need the results of the individual hardware implementation in terms of area A and frequency f of one single level of butterfly mapping with feedback with dependencies to N and p. As seen in Fig. 6, the area requirements are the same for any chosen type of FPGA, whereas the newer technology of the Virtex4 allows a higher clock frequency than for the VirtexE and Virtex2 family devices. The design area varies linearly with N and p, aside from small values of N and p where the amount of logic gates for the state machine controlling the comparator stages takes over the size of the network. The frequency however depends on the width p of the comparators and less on the density and width N of the network. This is mostly due to the fact that the EDA tools remap the design on the fixed internal architecture of the FPGA devices while trying to minimize the average propagation delay. For bigger values of N and p, the wiring over-influences the routing delays as the internal interconnects approach saturation, until the architecture cannot be laid out within the device area any more.

B. Evaluation Metric

Thompson's surveys [14], [15] on the VLSI area-time complexity for parallel sorting include various bitonic sorting schemes that demonstrate the existence of an $A \times T^2$ trade-off. In chip design, as the size of a circuit is determined as much by its inter-gate wiring as by the gates themselves, we consider the utilization of FPGA models for the hardware implementation of an efficient sorter based on the bitonic scheme as described in the previous sections.

To evaluate basic operations in the field of digital signal processing, several aspects have to be considered, e.g., the peak computational density [5] which represents the density of operations per unit of area-time (AT). In the communication domain, due to the importance of low-power operations, ATE_{ps} cost-functions can take into account the energy per output sample [7]. In our case, the challenge is to implement a system achieving the maximum throughput rate T^{-1} at a given input width N with a minimized silicon area A. With a combination of previous metrics, the cost function of the system architecture related to the width N of the sorter yields

$$\text{cost} = \frac{A \cdot t_\mathrm{s}\left(\frac{N}{M}, p_1\right)}{f_\mathrm{max} \cdot N}. \quad (6)$$

Regarding the complete system represented in Fig. 5, the total area A can be expressed with the local area A_i of each recurrent bitonic level i through

$$A = \sum_{i=1}^{m+1} 2^{m-i+1} \cdot A_i\left(\frac{N \cdot 2^i}{2M}, p_i\right), \quad (7)$$

with $M = 2^m$. The maximum frequency of the system is directly given by the worst synthesis result since it is not possible to determine the set of parameters that constraint the FPGA the most.

C. Architectural Comparisons

Originally depending on the overall length L of the keys, the internal partitioning with p_i at different levels implies the use of mapping registers, the size of which can be easily calculated. However, it has been experienced here that the length L of the keys sorted with this procedure does not play any role for the hardware implementation of the networks. Therefore it was set to 64 bits for the synthesis, giving a rather wide range of values for the parameter p_i. Hence Table I reports different parameter combinations regarding N, M and p_i considering an arbitrary minimum required throughput T^{-1}, whereas the p_i are adjusted to fit the best to any subdivision of the length of the keys L.

TABLE I
Experimental results with $N = 128$, $L = 64$ and $M = 4$.

T^{-1} (Keys/s)	p_1	p_2	p_3	f_max (MHz)	A (Slices)
500 M	64	16	32	82+	47 k
350 M	32	16	16	115+	21 k
220 M	16	8	8	144+	13 k

Furthermore, Fig. 7 shows the variations of the cost function (6) for two series of FPGAs from Xilinx against p and N. We obtain exactly the same behavior for the two models, whereas the newer generation yields as expected a relatively lower cost. However, it appears to be a optimum area-throughput trade-off related to the internal processing degree of parallelism reached for $p \cong 8, \forall\ N$. Actually, for $p \ll 8$, the throughput (T^{-1}) suffers from too many processing steps and for $p \gg 8$, the area A dominates in (6).

V. Conclusions and Outlook

This paper has presented the highly pipelined architecture of a synchronous sorting unit based on a multi-level recirculating bitonic merging network. On the one hand, we have saved area in using recirculation, and on the other hand, we had to duplicate the levels to increase the throughput. Hence, our work has demonstrated that a

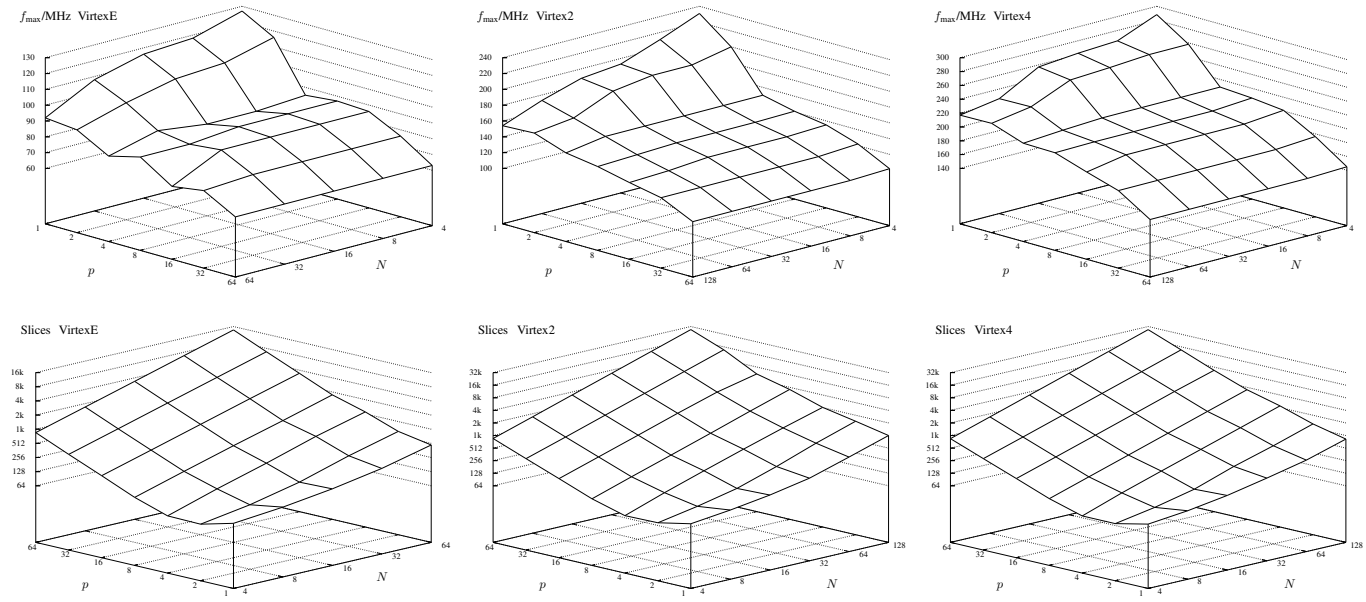

Fig. 6. Frequency (top) and area (bottom) against variations of N and p after synthesis on VirtexE (left), on Virtex2 (middle) and Virtex4 (right) FPGA devices. While the area of the design remains constant, the maximum frequency increases with the newer technologies.

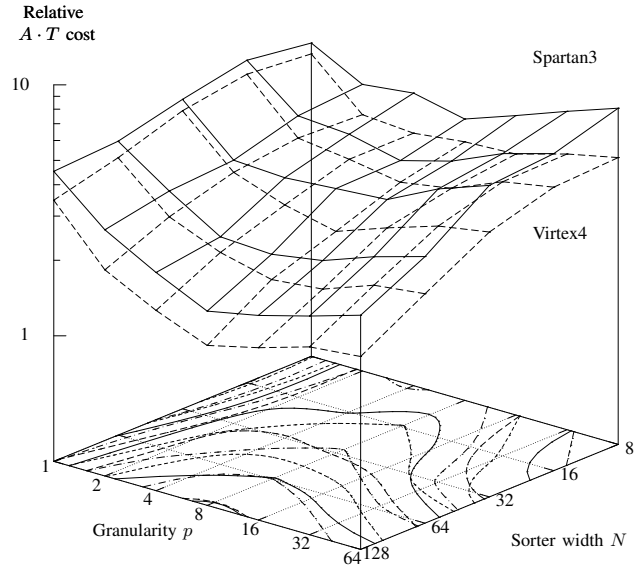

Fig. 7. Representation of the cost function for two families of FPGA against the width of the global sorter N and the internal granularity p for $M = 2$.

derived cost function can help balance a hardware design considering the resources necessary for a given application. Due to their specific intrinsic structure, FPGAs do not scale as well as standard VLSI circuits regarding area and clock frequency. It has been proved in [3] that LUT utilization is not always correlated with area efficiency. Moreover, FPGA design tools have the ability to make abstraction of a structural design of the butterfly network and place registers and LUTs on their two-dimensional grid in such a way that the routing delays are optimally distributed. Trying to understand the relation between algorithm modeling and architectural design on an FPGA basis, we exploited the structural retiming of an optimized bitonic sorter to provide the basis for the generation of IP cores as well as a hardware sorting paradigm for further research work.

REFERENCES

[1] K. E. Batcher, "Sorting Networks and Their Applications", *Proc. AFIPS Spring Joint Computer Conf.*, Vol. 32, pp. 307-314, 1968.

[2] K. Claessen, M. Sheeran, S. Singh, "The Design and Verification of a Sorter Core", *Proc. IFIP Conf. Correct Hardware Design and Verification Methods*, LNCS Vol. 2144, pp. 355-370, 2001.

[3] A. DeHon, "Balancing Interconnect and Computation in a Reconfigurable Computing Array", *Proc. ACM Symp. FPGAs*, pp. 69-78, 1999.

[4] A. DeHon, "Compact, Multilayer Layout for Butterfly Fat-Tree", *Proc. ACM Symp. Parallel Algorithms and Architectures*, pp. 206-215, 2000.

[5] A. DeHon, "The Density Advantage of Configurable Computing", *IEEE Computer*, Vol. 33, No. 4, pp. 41-49, 2000.

[6] S. Even, S. Muthukrishnan, M. S. Paterson, S. C. Sahinalp, "Layout of the Batcher Bitonic Sorter", *Proc. ACM Symp. Parallel Algorithms and Architectures*, pp. 172-181, 1998.

[7] H. T. Feldkämper, T. Gemmeke, H. Blume, T. G. Noll, "Analysis of Reconfigurable and Heterogeneous Architectures in the Communication Domain", *Proc. IEEE ICCSC*, pp. 190-193, 2002.

[8] D. E. Knuth, "The Art of Computer Programming, Vol. 3: Sorting and Searching", *Addison Wesley*, Sec. Edition, 1997.

[9] C. Layer, H.-J. Pfleiderer, "A Reconfigurable Recurrent Bitonic Sorting Network for Concurrently Accessible Data", *Proc. FPL Conf.*, pp. 648-657, 2004.

[10] J. D. Lee, K. E. Batcher, "Minimizing Communication in the Bitonic Sort", *IEEE Trans. Parallel and Distributed Systems*, Vol. 11, No. 5, pp. 459-474, 2000.

[11] S. Muthukrishnan, M. Paterson, S. C. Sahinalp, T. Suel, "Compact Grid Layouts of Multi-Level Networks", *Proc. ACM Symp. Theory of Computing*, pp. 455-463, 1999.

[12] T. Nakatani, S.-T. Huang, B. W. Arden, S. K. Tripathi, "K-Way Bitonic Sort", *IEEE Trans. Computers*, Vol. 38, No. 2, pp. 283-288, 1989.

[13] H. S. Stone, "Parallel Processing with the Perfect Shuffle", *IEEE Trans. Computers*, Vol. C-20, pp. 153-161, 1971.

[14] C. D. Thompson, "Area-Time Complexity for VLSI", *Proc. ACM Symp. on Theory of Computing*, pp. 81-88, 1979.

[15] C. D. Thompson, "The VLSI Complexity of Sorting", *IEEE Trans. Computers*, Vol. C-32, pp. 1171-1184, 1983.

[16] C.-H. Yeh, B. Parhami, E. A. Varvarigos, H. Lee, "VLSI Layout and Packaging of Butterfly Networks", *Proc. ACM Symp. Parallel Algorithms and Architectures*, pp. 196-205, 2000.