CMOS Realization of a Scalable High-Performance Binary Sorting Engine Suitable for Embedded Applications

Tugba Demirci
Sabanci University
Engineering and Natural Sciences
tugbad@su.sabanciuniv.edu

Ilhan Hatirnaz
Compaq Research
Laboratories, Shrewsbury, MA
ilhan@ece.wpi.edu

Yusuf Leblebici
Swiss Federal Institute of
Technology - Lausanne
yusuf.leblebici@epfl.ch

Abstract

The CMOS realization of a new scalable, modular sorting architecture is presented. The high-performance architecture is based on rank ordering, and on efficient implementation of multi-input majority (voting) functions. The overall complexity of the proposed bit-serial architecture increases linearly with the number of input vectors to be sorted (window size = m) and with the bit-length of the input vectors (word size = n), and the sorter architecture can be easily expanded to accommodate large vector sets. It is shown that the proposed sorting engine is capable of producing a fully sorted output vector set in (m+n-1) clock cycles, i.e., in linear time.

To demonstrate the concept, a full-custom sorting engine is realized to process 63 input vectors of 16-bits (m = 63, n = 16), using conventional 0.35 μm CMOS technology. The resulting sorter chip occupies a silicon area of 13 sqmm, operates at a clock frequency of 200 MHz, and it is capable of completing the sorting operation of 63 16-bit vectors within 78 clock cycles.

1. Introduction

The task of sorting an arbitrarily ordered vector set according to magnitude (either from-largest-to-smallest or from-smallest-to-largest) is one of the fundamental operations required in many digital signal processing applications. The solutions of many fundamental computer science problems like searching, finding the closest-pair, and frequency distribution etc., also depend on the efficient implementation of this function.

Sorting is an expensive operation in terms of area-time complexity; software-based solutions require word-level sorting and can become computationally intensive, while the overall complexity of hardware-based solutions usually increases very rapidly with the size of the input vector set (number of vectors) and with the bit-length of the input vectors [1], [2]. The design of efficient sorting engine architectures is therefore a significant challenge for overcoming the computational bottleneck of the binary sorting problem. A number of recent proposals for the realization of sorting networks rely primarily on median or rank order filters (ROF), yet their capabilities in terms of window size and bit-length are typically limited due to rapidly increasing hardware complexity [2], [3], [4].

In this paper, we present a new bit-serial sorting architecture based on rank-ordering. The hardware realization of this architecture results in a compact and fully modular sorting engine architecture that is capable of processing a large number of input vectors in linear time. The overall architecture is completely scalable to accommodate a wide range of window sizes and bit-lengths, and the hardware complexity only grows linearly with both of these parameters. The proposed sorting engine can also be used as an efficient building block in embedded SoC applications. To our knowledge, this represents the first demonstration of a linear-complexity sorter architecture on silicon. The sorting algorithm and the corresponding architecture are presented in Section 2 and Section 3. The full-custom CMOS realization of the majority function and the overall sorting engine architecture are discussed in Sections 4 and 5, followed by a summary of results.

2. The Sorting Algorithm

A bit-serial algorithm proposed in [5], [6] was chosen as the basis of the programmable sorter implemented in this work. In this algorithm, the problem of finding a rank-order-selection for n-bit long words is reduced to finding "n" rank-order-selections for 1-bit numbers.

The algorithm starts by processing the most significant bits (MSB) of the m = (2N + 1) words in the current window, through an m-input programmable majority function, to yield the MSB of the desired output. This output is then compared with the other MSBs of the window elements. The vectors whose MSB is not equal to the output have their MSB propagated down by one position, replacing the less significant bits of the corresponding words.

Thus, the bit-serial sorting algorithm operates with an input set (window) of "m" n-bit words. The output corresponds to a sequence of the input vectors in a desired rank order. Note that each bit-plane can be independently assigned its own rank value which is used to calculate the slice output. A detailed explanation of the basic sorting algorithm can be found in [6] and [7].

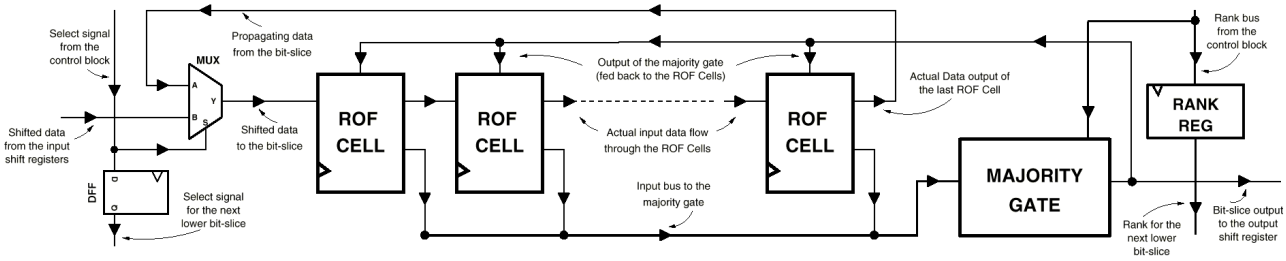

Figure 1: The block-level structure and the signal flow of one sorter bit-slice.

3. The Sorting Engine Architecture

The bit-serial operation flow of the algorithm described above suggests a simple bit-level pipelined data path architecture, consisting of data modifier-propagator blocks (ROF cells) to handle fine-grained data selection, and majority decision blocks (majority function) to determine output bits. The modular two-dimensional array architecture consisting of these two major blocks enables fully scalable construction of structures of arbitrary window size and bit-length. The bit-length dictates the number of the majority decision gates (rows), whereas the window size determines the number of ROF-cells driving one of these majority gates (columns). The structure of one row is shown in Fig. 1 and the circuit diagram of a ROF cell is shown in Fig. 2. Note that the circuit complexity of the ROF cell is fairly limited, resulting in a very compact realization.

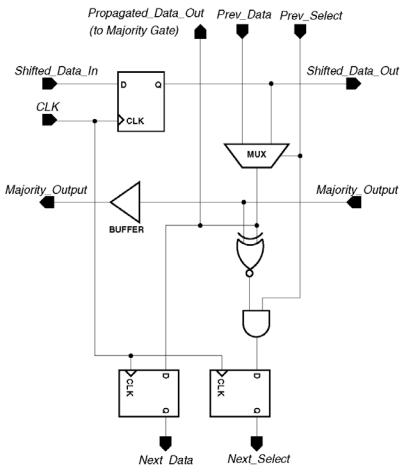

Figure 2: Gate-level circuit diagram of the ROF cell.

The complete sorter core has (n x m) ROF cells where m = (2N + 1) is the window size and n is the bit-length of the input words (vectors). Thus, it can be seen that the overall circuit complexity increases *linearly* with maximum window size (m) and with bit-length (n).

The proposed sorter architecture exploits the fact that the modular core described here is capable of generating one output vector per clock cycle, corresponding to the currently selected rank. If the ranking process is repeated on the same set of vectors instead of processing a continuous stream of new vectors, the members of the vector set can be sorted in linear time by simply changing (increasing or decreasing) the rank in each clock cycle. The circuit structure and the signal flow of one sorter bit slice that is designed to implement the bit-level operations described above will result in a regular, expandable structure, as seen in Fig. 1. The multiplexer on the input side is used for accepting the input vectors at the rate of one vector per clock cycle, as well as for circulating (rotating) the data until sorting is completed. The so-called "sorter core" is simply constructed by stacking "n" such bit-slices.

The overall architecture of the sorting engine is shown in Fig. 3. The flow of data through the modular ROF core is being regulated by complementary input and output shift registers, which are used to stagger the individual bit-planes of each input vector to enable bit-level pipelined operation. The control logic is responsible for regulating the data circulation path, and for applying the rank selection signals to the individual bit-planes, in ascending or descending order. The fact that each individual bit-plane is capable of processing a different rank at any given time significantly increases the overall efficiency of this architecture. In a typical sorting run, the control logic simply requests each bit-plane to process a different rank in each clock cycle, either beginning from the maximum rank and descending, or beginning from the minimum rank and ascending.

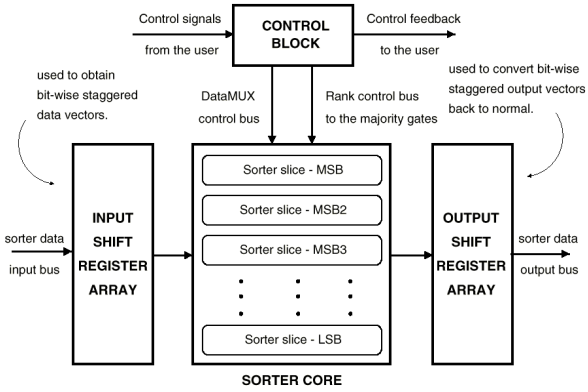

Figure 3: Top-level blocks in the sorter engine architecture. Note that each slice in the "sorter core" contains 63 ROF cells, one data MUX and one 63-bit majority block.

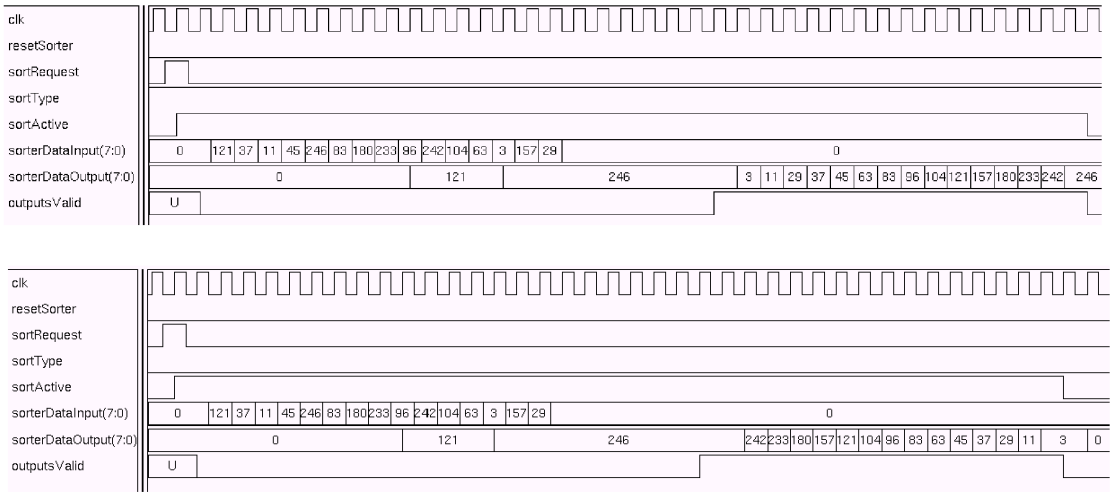

Figure 4: VHDL simulation results of the sorting architecture (m = 15, n = 8). The operation is done in ascending order (top) and in descending order (bottom) to demonstrate the function.

The proposed architecture has been described with VHDL to verify its operation. Fig. 4 shows simulated results of the sorting operation on an arbitrarily ordered set of 15 vectors (m=15), each with a bit-length of 8 bits (n=8). The user determines how many input vectors are to be sorted ("*actualWindowSize*", not shown in Fig. 4) and in which direction the sorting will occur ("*sortType*") and provides these inputs to the sorter block together with a request pulse ("*sortRequest*"). As soon as the request comes, the sorter block produces signal ("*sortActive*") which stays at the logic high level as long as the corresponding set of vectors is processed. It can be seen that the first output vector is generated with a latency of (n-1) clock cycles, after the last vector of the set is entered. The sorter block provides a signal to the user ("*outputsValid*") which goes high right at the last rising edge of the clock before the first vector is ready at the output ("*sortDataOutput*").

4. Realization of the Majority Function

The programmable majority (voting) function is the key operation that must be performed in each row. This function also determines the overall operation speed (i.e., the clock frequency), since a 63-input majority function must be performed in each row, during each clock cycle. Note that the other operations described in the previous sections only involve data transfers from one ROF cell to the next, thus, they do not represent a critical bottleneck in terms of the time budget.

The 63-input programmable majority block has been realized with a fully combinational parallel counter that consists of 57 full adders connected in a tree-network, and an output comparator network that consists of 27 basic logic gates. Overall, the worst-case logic depth of the entire majority block is equivalent to 8 full adders in cascade. Consequently, the input-to-output delay of the programmable majority function is smaller than 4.5 ns. This allows a maximum clock frequency of 200 MHz for the entire system.

The layout of the 63-input adder-tree network and the comparator block is shown in Figure 5. The silicon area occupied by this block is (208 µm x 380 µm), which is much smaller than the total area occupied by the ROF cells of the corresponding row. Also note that the classical realization of the 63-bit majority function would require an equivalent of 63 6-bit adder circuits, arranged in a network of a logic depth of 64 (synthesized from HDL description). The post-layout simulation results of the 63-bit majority network are shown in Figure 6.

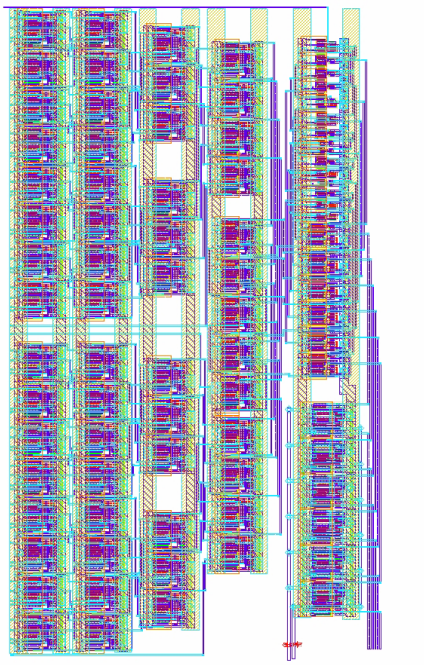

Figure 5: Full-custom layout of the 63-bit programmable majority block, consisting of a tree network with 57 full adders and the comparator. The total area is (208 µm x 380 µm).

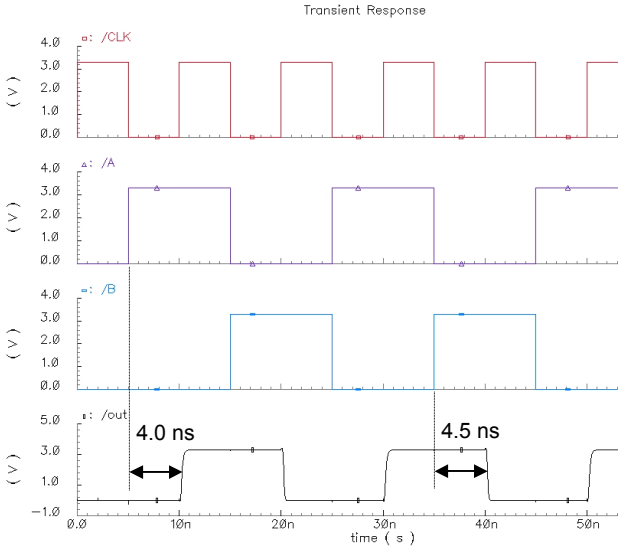

Figure 6: Simulation results of the 63-bit programmable majority function block, where 32 inputs are assigned logic-1 and 31 inputs are assigned logic-0 (alternating). The worst-case propagation delay is smaller than 4.5 ns.

5. Realization of the Sorter Architecture

The binary sorting engine architecture designed to process 63 input vectors of 16-bits (m = 63, n = 16) has been realized using conventional 0.35 µm CMOS technology. The architecture consists of 16 rows, where each row is capable of processing 63 bits simultaneously. To reduce signal propagation paths and to simplify a balanced clock distribution, the rows were designed with a folded geometry, and the 16 rows were divided in two main columns, with 8 rows each. The top level layout of the chip is shown in Figure 7. A four-level balanced clock buffer network was used to distribute the system clock to minimize skew. One 63-bit cell row and one 63-bit majority network are also highlighted in the layout. With its highly regular architecture and compact size, this block would be a suitable candidate for embedded applications where the sorting function is required.

6. Conclusion

In this paper, we present a highly modular architecture for the realization of high-speed binary sorting engines. The architecture consists of (i) a regular "core" array that is completely scalable to accommodate large window sizes and bit-lengths, (ii) input/output shift registers, and (iii) control logic to regulate the bit-level processing of data. It was shown that the complexity of the proposed bit-serial pipelined architecture increases *linearly* with the number of input vectors (m) to be sorted, and with bit-length of the input vectors (n). It was also demonstrated that the proposed sorting engine is capable of producing a fully sorted output vector set in (m+n-1) clock cycles, i.e., in *linear* time.

A full-custom sorting engine chip was realized to process 63 input vectors of 16-bits (m = 63, n = 16), using conventional 0.35 µm CMOS technology. The resulting sorter chip operates at a clock frequency of 200 MHz, and it is capable of completing the sorting operation of 63 16-bit vectors within 78 clock cycles. To our knowledge, this represents the first demonstration of a linear-complexity sorter architecture on silicon.

References

[1] P. Wendt et al., "Stack filters", *IEEE Trans. Acoust., Speech, Signal Processing*, pp. 898-911, 1986.

[2] W.K. Lam and C.K. Li, "Binary sorter by majority gate", *IEE Electronic Letters*, Vol. 32, July 1996.

[3] B.K. Kar, D.K. Pradhan, "A new algorithm for order statistic and sorting", *IEEE Trans. on Signal Processing*, vol. 41, pp. 2688-2694, August 1993.

[4] C.C. Lin, C.J. Kuo, "Fast response 2-D rank order algorithm by using max-min sorting network", *International Conference on Image Processing 1996*, Vol. 1, pp. 403-406.

[5] I. Hatirnaz, F.K. Gurkaynak, Y. Leblebici, "A modular and scalable architecture for the realization of high-speed programmable rank-order filters", *ASIC/SOC'99 Proceedings*, pp. 382-386, 1999.

[6] I. Hatirnaz, Y. Leblebici, "Scalable binary sorting architecture based on rank ordering with linear area-time complexity", *ASIC/SOC'00 Proceedings*, September 2000.

[7] I. Hatirnaz, F.K. Gurkaynak, Y. Leblebici, "A compact modular architecture for high-speed binary sorting," *ISCAS-2000 Proceedings*, May 2000.

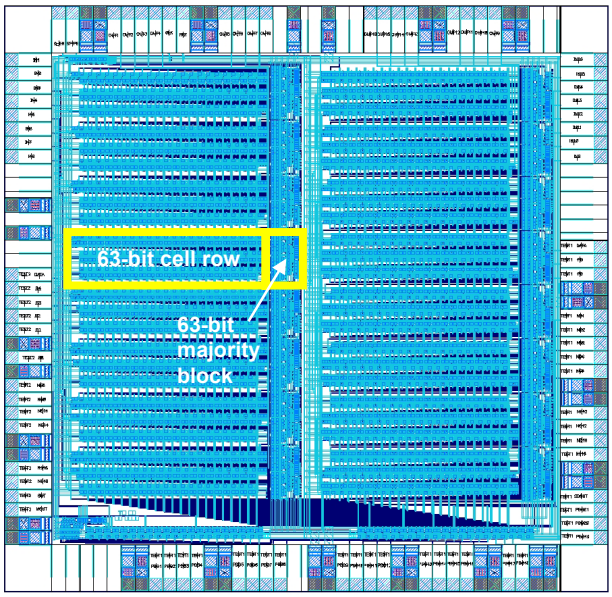

Figure 7: Top-level layout of the sorter chip, realized with 0.35 µm CMOS technology. The core area is 13 sqmm.