HIGH SPEED WITH LOW-POWER DATA BASE SORTING UNITS

Pradeep P,
Department of Electronics and Communication,
Karunya University,
Coimbatore, India.

J Samson Immanuel,
Department of Electronics and Communication,
Karunya University,
Coimbatore, India.

Abstract— **Sorting is an important requirement that deals with large amount of data base applications. This work represents efficient techniques for designing high speed sorting units and our proposed sorting units are optimized for large data base systems in which only M largest numbers from N inputs are required, because this is a frequently occurring situation in applications for data mining, digital signal processing, network processing, and VLSI design. We propose an iterative sorting technique for largest data sets by utilizing a smaller sorting modules. This work shows analysis various sorting units that subjects when length of input increases their resource parameters scale linearly, their delay scales along logarithmically, and frequencies remains nearly constant.**

Index Terms— **Data mining, digital signal processing, iterative sorting, VLSI design.**

I. Introduction

Sorting is an important operation in a wide range of data processing applications that includes data mining, databases, digital signal processing, network processing, VLSI design, scientific computing, searching, scheduling, and pattern recognition. For applications that need very high-speed sorting, hardware sorting units are implemented using either ASICs or FPGAs to meet their performance requirements. Based on target applications, hardware sorting units vary greatly not only in architecture but also in the number of inputs, width of inputs and its pipeline depth that they can process.

In many applications, only the M largest (or smallest) output values need to be selected from a total of N input values, where M < N. In signal processing applications, for example only the M strongest signals or M nearest points in space are observed to be analyzed. In data mining, searching, and database scheme, only the top objected outputs that records the most with respect to a given input that may require further processing steps. We refer to units that only return the M largest (smallest) outputs, but do not guarantee that these M outputs are sorted, as max (min)-set-selection units. We refer to units that only return the M largest (smallest) outputs in sorted order as partial sorting units. Our work targets on the design of max-set-selection units and partial sorting that return the $M = 2^m$ largest values from $N = 2^n$ inputs, where m and n are whole numbers and $1 < M < N$. We refer to these units as N-to-M max-set-selection units and partial sorting units.

The main concepts of this paper and [14] are:

1. Modular techniques for designing N-to-M partial sorting and max-set-selection units. These units are composed of small and regular building blocks that are connected in a modular design, and thereby reducing verification processing delay and shortening the design process. Our designs have low delay, high speed, and modest resource requirements that can be pipelined easily and have parameterized pipeline depth and data widths, and scale well to large values of N and M.
2. The analysis of sorting and maximum -set-selection units that holds theoretical and synthesis estimates of our units delay, processing speed and resource requirements. This analysis indicates that for a given number of outputs, resource elements scales linearly with number of inputs, delay scales logarithmically with the number of inputs, and the frequency remains almost constant.
3. Compared to other sorting units, our work enhances all inputs in sorted order, our N-to-M partial sorting and maximum-set-selection units have lower delay and area.

II. Methodology

The basic storing block contains two bits, the corresponding bits of each word. We use two D-flip-flops with three multiplexers. The signal *sel* decides which bit will be updated and put in the output port when shifted ahead [4]. The remaining bit is fed back to the same flip-flop without changing its state. The output port of this block is connected to the input port of the subsequent block. The registers are an array of the blocks such that both stored words are interleaved. A continuous data stream can be, for instance,

Pradeep P, Department of electronics and communication Engineering, Karunya University, Coimbatore, India.
J Samson Immanuel, Department of electronics and communication Engineering, Karunya University, Coimbatore, India.

the data generated by a measuring instrument in which the tags are the time stamps of every measure. In this case, the data stream is naturally sorted. But suppose we merge several data streams coming from different sets, each of them with different random latencies, then the merged data stream will be already partially sorted.

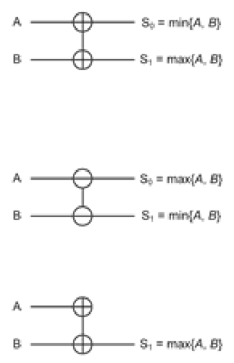

Fig. 1 schematic symbol of basic building blocks for sorting networks.

Sorting Networks have been used for sorting large problems using vector operations, for example, using the Cell Broadband Engine Architecture or GPUs. On FPGAs, sorting networks are typically applied to smaller problem classes, for example, to compute a median value. For large problems, sorting networks require greater I/O throughput as they require more sort keys and produce a huge amount of result data at the same time. This leads to alternating operation between I/O bursts and processing bursts of the design.

We use N-to-M partial sorting networks that is presented and theoretically analyzes sorting algorithms to proceeds finding/sorting the M largest values from N inputs and then design scalable architectures based on the proposed algorithms. Our N-to-M partial sorting networks have very less delay when compared to other sorting designs when outputs only M largest values and our N-to-M max- set-selection units further decrease the delay and resource requirements by not producing their outputs in sorted order. Our sorting units aims for the applications that crave very less delay sorting. Our iterative units target applications that needed moderate-delay sorting by trading increased delay for reduced area and I/O bandwidth.

A. Even-Odd merge sorting

An even-odd merge sorting network merges two ascending sequences of length L=2 for a sorted sequence of length L. Each even-odd merge sorting network is composed of a number of even-odd merging units. An L-input even-odd merging unit (EOM-L) merges two ascending input sequences in to a single ascending output sequence. It contains $\log_2$ (L) CAE (Compare and exchange) stages, where each stage has between L=4 and L=2 CAE blocks. An EOM-L takes two length L=2 ascending sequences, A and B. The EOM-L merges the input values having even indices in A with the input values having even indices in B, and also merges input values in A and B having odd indices. The result is a sorted sequence of values with odd indices (S_E) and a sorted to the sequence of values with even indices (S_O).

B. Bitonic sorting

Each bitonic sorting network is composed of a number of bitonic merging units to merge bitonic sequences. Linking together an ascending and a descending sequence forms a single bitonic sequence. Usually bitonic sorting network inductively merges a descending and an ascending sequences each of length N=2 to make a sorted sequence of length N.

An L-input bitonic merging unit (denoted as BM-L) contains $\log_2$ (L) stages of parallel CAE blocks, where each stage corresponds to a CAE stage with L=2 CAE blocks. Therefore, a BM-K requires $\log_2$ (L) L=2 CAE blocks. For instance, an increasing BM-8 unit has three CAE stages and requires 3 x 8/2 = 12 CAE blocks. A BM-K unit is itself composed of a level of L=2 parallel CAE blocks followed by two parallel BM-(L/2) units. The +ve BM-8 is designed from a level of four parallel CAE blocks followed by two parallel BM-4 units that have four CAE blocks each. The difference in the number of CAE blocks between bitonic and even-odd merge sorting units is given by $[2^{n-1} \times (n - 2) + 1]$, showing that the difference in the number of CAE blocks increases linearly with the number of inputs.

C. Max-Set-Selection units

1. 4-output Max-Set-Selection units

Here we focus on 2^n-to-4 max-set-selection units (where n=3). To design 2^n-to-4 max-set-selection units, we take use of the fact that only four largest inputs are needed, in no particular order, to reduce resource parameter requirements and number of CAE stages.

An N-input, N-output bitonic or even-odd merge complete sorting unit is composed of $[(\log_2 N) \times (\log_2 N + 1)/2]$ CAE stages, where $N = 2^n$.

The total number of CAE blocks in an N-input bitonic sorting unit is $[N \times \log_2 (N) \times (\log_2 (N) + 1)/4]$.

An N-input even-odd merge sorting unit has $[N/4 \times (\log_2 (N)) \times (\log_2 (N) - 1) + N - 1]$ CAE blocks.

Bitonic sequences has a special characteristic property of computing Max/Min operations for two sorted sequences (one decreasing and another increasing) each of L numbers. This generates two new sequences of L numbers in which numbers in one sequence are lesser than numbers in another sequence. In BM-L units, the first level of parallel CAE blocks partitions the input numbers into two (L=2) number subsets. The larger numbers and the smaller numbers. Although, the initial level of parallel CAE blocks in EOM units need to be wired again for generating correct larger and smaller subsets of numbers. In our design we use always A<B in order to reduce the complexity in computing

large amount of data sets, which leads to reduced power and area domains.
The above mentioned logarithmic notations ensures the reduced timing for the design.

TABLE 1

The Enforced Number of CAE Stages and CAE Blocks being used for 2^n-Input Bitonic and Even-Odd Merge Sorting Units

# of inputs and outputs ($N=M=2^n$)	# of CAE stages	# of CAE blocks		
		Even-Odd	Bitonic	Difference
08	06	19	24	05
16	10	63	80	17
32	15	191	240	49
64	21	543	672	129
128	28	1471	1792	321
256	36	3839	4608	769

The above table clearly shows the resource requirements for even-odd merge and bitonic sorting units and gives a clear view of the area that has to be optimized for the processing of larger data sets.

2. *2^n-to-4 Max-Set-Selection Units*

The below mentioned figure shows architecture of 2^n-to-4 max-set-selection unit that use bitonic and even-odd merge algorithms, respectively, we consider here n<3. This sorting design illustrates the OEM-8 unit with a level of Max units with wirings that differ from the first level of parallel CAE blocks in the OEM-8 unit [14]. These enhances by decreasing the required number of CAE stages from 6 in 8-input sorting units to 4 in 8-to-4 max-set selection units.

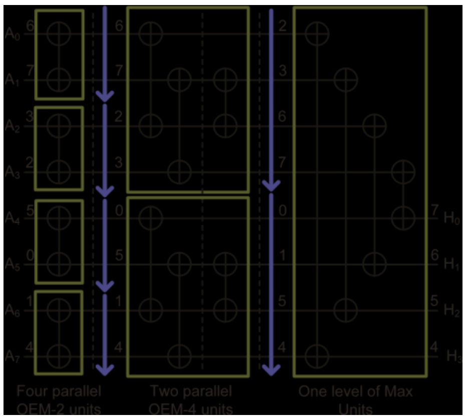

Fig. 2 The CAE network for 8-to-4 even-odd merge max-set selection unit.

In the above figure there are total 14 compare and exchange blocks which are chosen in such away that the even-odd merge max-set selection is processed from four CAE stages. This network basically contains series of CAE stages and CAE blocks that are processed in parallel fashion. This parallel design of CAE blocks undergoes sorting very fast than the O(N x log(N)) time acheivable than speeder sequential algorithms based on software thesis. Here O indicates the run time of the algorithm.

The sorted outputs are obtained as H0, H1, H2 and H3 in descending order [13]. The arrows pointing downwards indicates that the sorting of input data is processed in ascending order.

The output obtained is a sorted sequence of four largest out comes from the 8 inputs.

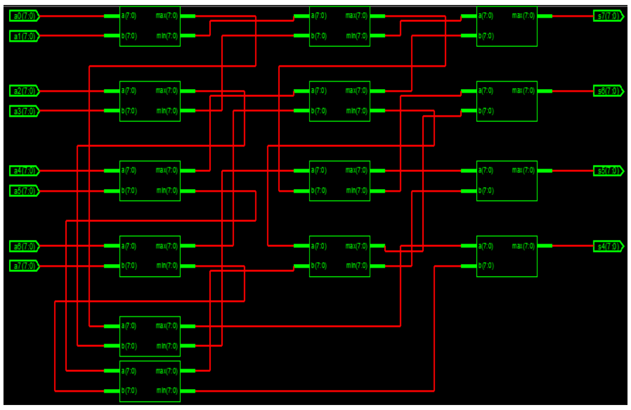

Fig. 3 RTL schematic for 8-to-4 even-odd max-set selection unit

The RTL schematic of 8-to-4 even-odd max-set selection unit and the port polarity of the schematic is considered as output architecture.

D. *Iterative Max-Set-Selection units*

This work mainly enhances on below mentioned processes

1) Our designs are upgraded to compact precisely M outputs from N inputs are needed.
2) By getting the relevant number of input values in each cycle this work avoids using increased storage for intermediate memory blocks.
3) To achieve improved area and processing speed of the sorting units, this design instead of using complete sorting units, make use of iteratively max-set selection units.

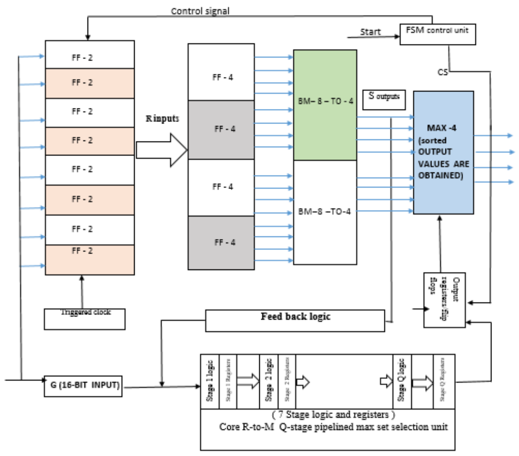

Fig. 4 Iterative max-set-selection unit

From the above figure the iterative max-set selection uses R-to-S even-odd max-set selection or otherwise bitonic units of different individual pipeline depths of Q values. In our design FSM is used to propagate through three sequential execution states. These stages are namely warm up state, steady state, and completion state. Along with input registers, the delay of this iterative max-set selection designs in the form of triggering clock cycles and the delay of our design which is enclosed by

$$\text{Delay} = \left\lceil N/G \right\rceil + \left\lceil (Q+1)^2 \times S / (G+S) \right\rceil + Q + 1$$

Where G is the number of new input values which are acquired in each clock cycle from core max-set selection unit, Q is the pipeline depth, S is the number of outputs achieved from core max-set selection unit, N is over all number of input values.

From Fig. 4 in the first stage the G input values access and mount to generate over core max-set selection unit pipeline block, and accomplish before the occurrence of mean intermediate output values. The steady state begins by accessing of first set of input values along the core max-set selection unit by following output data. In the mean while for every clock cycle of steady state, a group of G fresh input values obtained at the input and a set of S intermediate output values are generated and instantly consumed by the core max-set selection unit, where S = R – G.

In this stage, the intermediate output values are fed back to Core max-set selection unit for sorting of newly obtained input values. The third stage called completion stage starts when all input values are obtained through core max-set selection unit. This work address some techniques that can be drawn-out to cover a wide range of sorting and the overall max-set-selection units. The smallest values are ignored in the earlier stages to reduce resources and delay. When the intermediate result values are sorted at the inputs of the sorting unit and the values are generated through core max-set-selection unit. If once R input values are sorted, then all tis values are freely proceeded to core max-set selection unit. This whole iterative process is terminated for a final max-set selection run with R or otherwise with some remained valid input values that results in producing final S outputs.

III. SIMULATION RESULTS

We synthesized our proposed designs to a Virtex-5 (speed grade -2) using Xilinx Asics Synthesis Technology. Table 2 shows logic description constraints results for 2^n-to-4 bitonic max-set-selection units. The assumed max-set-selection units can achieve high frequencies due to the regular pipelined structure for our designs.

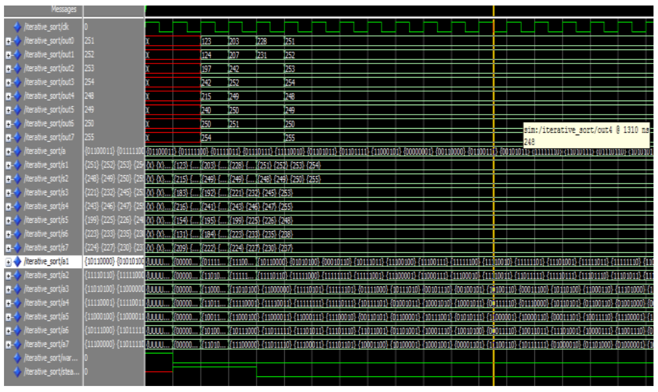

Fig 5 The Simulated output wave form of 256-bit iterative max-set selection unit.

The Fig. 5 shows 256-bit iterative max-set selection which is simulated by using Xilinx vertex5. The clock signal which is used in our work effectively performs the counting of data sets and outputs the desired values in ascending or descending order. Here we are using count 0, count 1, and count 2, for 256-bit data sets that can be sorted in three different stages. The whole process terminates by FSM signal which is preloaded with the signed/unsigned input values. In each cycle our design avoids using intermediate values or additional memory storage by giving the appropriate number of inputs to the design. Thus it increases the speed of the design and reduces the area complexity.

The proposed designs are synthesized and implemented by using the Asic compiler library with setting the property as vertex-5 and selected device as 6vl-760ff1760-2. These designs are pipelined and the outputs are registered in the Fig.5 and Fig.6. For all the synthesis results, the parameterizable data width taken as the CAE width, which can be easily changed, that is set to ten unsigned bits. All max-set-selection units can achieve high frequencies due to the regular pipelined structure of the designs.

The Fig.6 clearly shows the simulated wave form that indicates sorting design from Xilinx vertex 5 for n=3 in 2^n-to-4 max-set selection unit. The speed of this unit is quite quicker for the data to be sorted. This smaller building blocks that ensure good processing speed is further used for large data sets which consists of large data to be sorted in many applications.

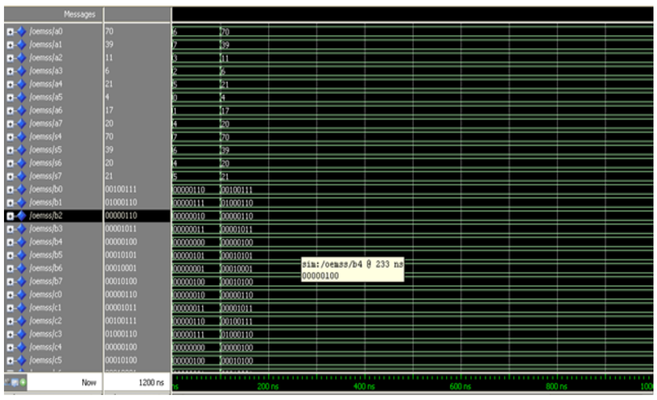

Fig.6 The Simulated output wave form of 2^n-to-4 max-set selection unit for n=3.

The table 2 draws out the logic description of our designs and enables the cost parameters analysis in various aspects. The high speed, low delay sorting units are widely used in many applications where the large amount of data is to be sorted in order to decrease the complexity of work processes. For small fraction of resource requirements, the proposed iterative max-set selection data sorting unit could provide reasonable speed and delay for certain target applications. The power consumed is very low that is the ultimate aspect of this designs.

TABLE 2

Table shows logic description and design cost parameters utilized for this work.

Design Constraints	Area				Speed grade	Time (ns)	Feq (MHZ)	Pwr(W)
	# LUT FF pairs used	Fully used pairs	Unique control sets	# of bonded IOBs				
Iterative max-set selection	243	504	0	0	-2	0.65	46.154	4.475
2^n-to-4 max set selection	14219	504	05	65	-2	14.39	14.143	4.447

IV. CONCLUSION

From the results obtained its concluded that the proposed designs effectively ensures the easy way to achieve the most of amazing facts about technological knowledge in sorting of data sets in many applications. The ASIC tool is well used to simulate our design which are adopted for ISE simulator. The iterative max-set selection Used in this work enables high processing speed by allowing to utilize low power which is the major advantage.

V.ACKNOWLEDGMENT

Extreme gratitude is conveyed to all the staff members of our department for extending their helping hands to make this project a success.

VI. REFERENCE

[1] B. Ahn and J.M. Murray, "A Pipelined, Expandable VLSI Sorting Engine Implemented in CMOS Technology," Proc. IEEE Int'l Symp. Circuits and Systems, pp. 134-137, May 1989.

[2] C. D. Thompson, "The VLSI complexity of sorting," IEEE Trans. Com- puters, vol. C-32, pp. 1171–1184, Dec. 1983.

[3] C.J. Kuo and Z.W. Huang, "Modified Odd-Even Merge-Sort Network for Arbitrary Number of Inputs," Proc. IEEE Int'l Conf. Multimedia and Expo, pp. 929-932, Aug. 2001.

[4] C. Layer, D. Schaupp, and H.-J. Pfleiderer, "Area and Throughput Aware Comparator Networks Optimization for Parallel Data Processing on FPGA," Proc. Int'l Symp. Circuits and Systems, pp. 405-408, May 2007.

[5] C. Layer and H.-J. Pfleiderer. A Reconfigurable Recurrent Bitonic Sorting Network for Concurrently Accessible Data. In Proceedings of the International Conference on Field Programmable Logic and Applications (FPL), pages 648–657, 2004.

[6] D. Koch and J. Torresen, "FPGASort: A High Performance Sorting Architecture Exploiting Run-Time Reconfiguration on FPGAs for Large Problem Sorting," Proc. Symp. Field Programmable Gate Arrays, pp. 45-54, 2011

[7] E. Herruzo, G. Ruiz, J.I. Benavides, and O. Plata, "A New Parallel Sorting Algorithm Based on Odd-Even Mergesort," Proc. Int'l Conf. Parallel, Distributed and Network-Based Processing, pp. 18-22, 2007.

[8] J. Ortiz and D. Andrews, "A Configurable High-Throughput Linear Sorter System," Proc. Symp. Parallel Distributed Processing, pp. 1-8, 2010.

[9] J.C.R. Bennett, D.C. Stephens, and H. Zhang. High speed, scalable, and accurate implementation of packet fair queueing algorithms in ATM networks. In ProceedingsofIEEEICNP'97, pages 7{14, Atlanta, GA, October1997.

[10] J.P. Agrawal, "Arbitrary Size Bitonic (ASB) Sorters and Their Applications in Broadband ATM Switching," Proc. IEEE Int'l Conf. Computers and Comm., pp. 454-458, Mar. 1996.

[11] K.J. Liszka and K.E. Batcher, "A Generalized Bitonic Sorting Network," Proc. Int'l Conf. Parallel Processing, pp. 105-108, Aug. 1993.

[12] Lin, C.S., Liu, B.D.:Design of a Pipelined and Expandable sorting Architecture with Simple Control Scheme. IEEE International Symposium on Circuits and Systems, Volume: 4, pp. 217–220, 26-29 May. Scottsdale, Arizona, USA (2002)

[13] M. Ajtai, J. Komlo´s, and E. Szemere´di, "An O(n Log n) Sorting Network," Proc. Ann. ACM Symp. Theory of Computing, pp. 1-9, May 1983.

[14] Michael J. Schulte , Amin Farmahini-Farahani, Henry J. Duwe III, and Katherine Compton, "Modular Design of High-Throughput , Low-Latency sorting Units" IEEE transactions on computers, Vol.62, No.7, July 2013.

[15] M.F. Ionescu and K.E. Schauser, "Optimizing Parallel Bitonic Sort," technical report, Univ. of California at Santa Barbara, 1997.

[16] N. K. Govindaraju, J. Gray, R. Kumar, and D. Manocha. GPUTeraSort: High Performance Graphics Coprocessor Sorting for Large Database Management. In Proceedings of the ACM international conference on management of data (SIGMOD), pages 325–336. ACM, 2006.

[17] R. Marcelino, H.C. Neto, and J.M.P. Cardoso, "Sorting Units for FPGA-Based Embedded Systems," Proc. IFIP

Cong. Distributed Embedded Systems: Design, Middleware and Resources, pp. 11-22, Sept. 2008.

[18] Ratnayake, K., Amer, A.: An FPGA Architecture of Stable-Sorting on a Large Data Volume : Application to Video Signals, 41st Annual Conference on Information Sciences and Systems, pp. 431–436, 14-16 March, Baltimore, USA (2007)

[19] R. Cole and A. R. Seigel, "Optimal VLSI circuit for sorting," J. ACM, vol. 35, pp. 777–809, 1998.