Area Efficient Low Latency Sorting Unit Design Using Scalable CMOS Comparator

R. Guru Deepthi
PG Scholar (M.E VLSI Design)
Nandha Engineering College
Erode.
divvdeeps@gmail.com

K.Sharmilee
Associate Professor
Nandha Engineering College
Erode.
sharmijaisha@gmail.com

V.Bindhya
PG Scholar (M.E Applied Electronics)
Nandha Engineering College
Erode.

M.Manimegala
PG Scholar (M.E VLSI Design)
Nandha Engineering College
Erode.

Abstract **- Sorting is an important technique used in many applications such as visual processing unit (VPU), Digital Signal Processing (DSP), network processing etc. To achieve high throughput rates today's computers perform several operations simultaneously. This paper presents the efficient technique for low latency area efficient sorting units. The sorting unit utilizes parallel sorting method which uses compare and exchange blocks. Two popular parallel sorting algorithms are Bitonic sorting network and Odd even sorting network. Using these two sorting networks, it can be shown that the area can be reduced with latency. When input increases, the number of blocks also increases and hence the area increases. To overcome this problem, iterative sorting unit can also be used, in which the inputs are given under looping process. In cases in which I/O bandwidth or area is limited and latency requirements are not stringent, a small max-set-selection unit can be employed using iterative process to obtain the largest values from a given input data set. Such iterative max-set-selection units can provide throughput, latency, and resource requirement tradeoffs. Also to obtain efficient area, the compare and exchange block used in parallel sorting is modified using CMOS comparator. The operation is mainly based on Mux logic and some basic logic gates.**

Keywords **- - Bitonic sorting, latency, odd even sorting, parallel sorting, throughput**.

I. INTRODUCTION

In the past, the major concerns of the vlsi were area, performance, cost and reliability. The sorting problem has been investigated under various parallel architectures, since utilizing many functional units to sort concurrently can improve performance. Based on the target applications, hardware sorting units vary greatly not only in architecture but also in number of inputs and width of the inputs they process. In order to achieve high throughput rate, today's computer perform several operations simultaneously. Not only input operations performed concurrently with computing, but also in multi processors several computing operations are done concurrently.[3] In previous research of sorting units, it must produce all of its input in sorted order. But in most of the applications, all the given inputs need not to be sorted. For example, in HEP applications, only the M most energetic particles are considered. Similarly in digital signal processing applications, only the M strongest signal need to be analyzed. This paper focuses on partial sorting and max-set-selection units that discard small inputs as early as possible to reduce the sorting units latency and hardware complexity. Batcher introduced the bitonic sorting network and the odd-even sorting network which can sort N keys in O(log2 N) time, and with O(N log2 N) comparators. [7]

A sorting network is a collection of interconnected compare-and-exchange (CAE) block. Sorting networks consist of comparators, which are in fact hardware implementations of the CE operation. A comparator has two inputs and two outputs. Depending on the sense of ordering, comparators can be of two kinds as shown in Fig 1 (a), (b).

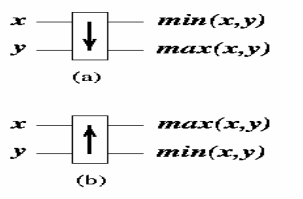

Fig 1. (a) The default type of comparator (b) Second type of comparator

Fig 2 is similar to a multistage interconnection network, except that the building block is a comparator rather than a *2×2* switch. Unsorted input sequence $X=x_1,...,x_N$ placed on input wires of the leftmost column of comparators passes through the network so it becomes sorted output sequence $Y=y_1,...,y_N$ on output wires on the right most column.

The way how columns of comparators are interconnected determines the sorting algorithm. If N is greater than the number K of input wires of the sorting network, then we assume that the input data set is split into N/K subsequences, which are then sorted. Comparators are equipped with buffers for N/K numbers and instead of CE operation it performs compare and split(CS) operation in which two input sequences are merged and split again to lower and upper half.

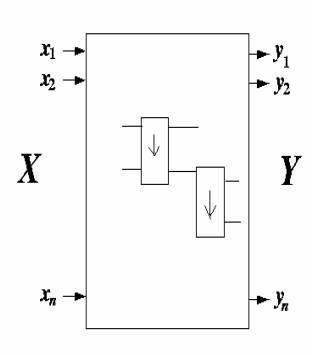

Fig 2. Sorting network composed from columns of basic comparators.

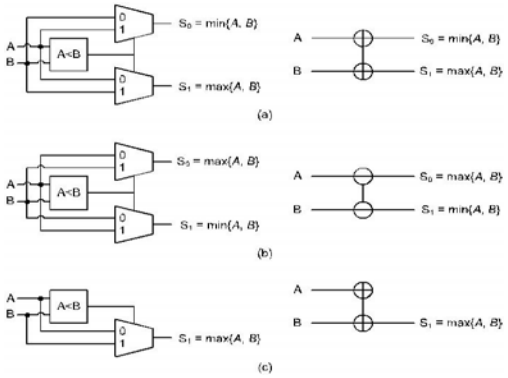

Fig 3. The high level implementation(left) and its symbol

II. BITONIC SORTING NETWORK

A bitonic sequence is one, which consists of two subsequences, one that monotonically increasing and the other monotonically decreasing. The bitonic sorting is designed particularly for parallel machines. The bitonic sort produces the bitonic sequence by sorting the two halves of the input sequence in opposite directions. Each bitonic sorting network is composed of bitonic merging units. This sorting network consists of O(n $\log_2$ n) comparators and have an delay of O($\log_2$n), where n is

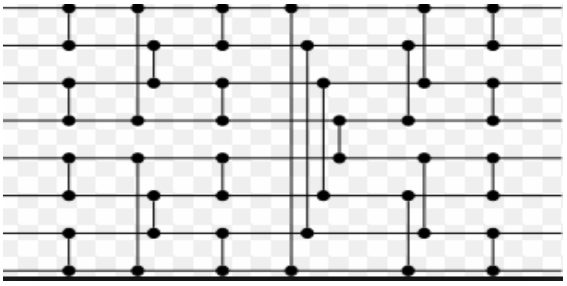

Fig 4. Flow of Bitonic sorting

the number of items to be sorted. A K input bitonic merging unit (denoted as BM-K) requires $\log_2(K) * K/2$ CAE blocks.[7]

An 8 input bitonic sorting unit has four parallel BM-2 units, two parallel BM-4 units, and one BM-8unit. Assuming the unit is pipelined so each stage takes one clock cycle, it can generate the sorted outputs in six cycles and can begin a new sort each cycle.

III. ODD EVEN SORTING NETWORK

Batcher's odd–even merge sort is a generic construction 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. It recursively merges two ascending sequences of length K/2 to make a sorted sequence of length K.. A K input odd-even merging unit is represented by OEM-K. This merges the input values having odd indices in A with the input values having odd indices in B. similarly it merges the even indices also.

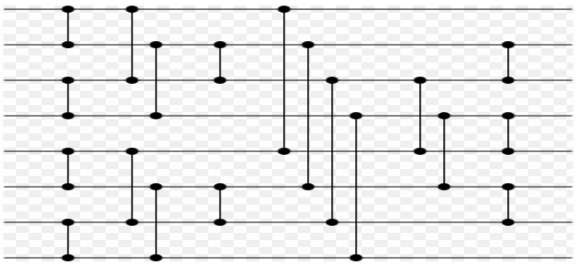

Fig 5. Flow of odd even merge sort

The result is a sorted sequence of values with even indices (S_E) and a sorted sequence of values with odd indices(S_E). in the final stage, the S_O and S_E are merged to form a single sorted output sequence K. [7]

IV. PARTIAL SORTING AND MAX SET SELECTION UNIT

In many applications, it is not necessary to return all of the sorted inputs. Applications often only need to determine the M largest or M smallest numbers from N inputs, where M < N and M and N are both integer powers of two ($M = 2^m$, $N = 2^n$). Partial sorters provide the 2^m largest values in sorted order, and max-set-selection units provide the 2^m largest values in arbitrary order. Partial sorters and max-set-selection units are key components in many applications. In multimedia applications, partial sorters speed up data sorting algorithm. Moreover, auxiliary max-set-selection units can cooperate with general-purpose processing units in embedded and database management systems to accelerate data search and sort algorithms. In cases such as this, Batcher's algorithms can be optimized to generate only the 2m largest numbers from 2n inputs with less latency and fewer CAE blocks than a complete sorting network.[7] This partial sorting and max set selection unit can be extended to

- 4- output max set-selection and partial sorting units
- 2^n-to-2^m max-set-selection and partial sorting units

V. PROPOSED SCALABLE CMOS COMPARATOR

The comparator used in the existing system reduces the area to some extent. But to further reduce the area, this comparator is replaced with the scalable cmos comparator. It performs efficiently when compared to the previous one. Here the in depth of each operation is analysed here. By using these components, the CMOS comparator for 8 bit can be built. These are the major modified components with which the bitonic and odd even merge sort can be constructed efficiently. Here each symbol represents a given functionality. The symbols of the logic gate and the corresponding fan in and fan out are given in the table 1.

TABLE I
LOGIC GATE REPRESENTATION FOR SYMBOLS USED IN FIG 6

Symbols (Cells)	Logic Gate	Maximum Fan-in/Fan-out And (Transistor Counts)
		2 / 4 (12)
		4 / 4 (8)
		5 / 1 (20)
MUX-Logic	TG: Transmission Gate	3 / 2 (12)

The XNOR gate is a digital logic gate whose function is the inverse of the exclusive OR (XOR) gate. The two-input version implements logical equality, behaving according to the truth table to the right. A HIGH output (1) results if both of the inputs to the gate are the same. If one but not both inputs are HIGH (1), a LOW output (0) results. If no specific XNOR gates are available, one can be made from four NOR gates or five NAND gates in the configurations shown below. In fact, any logic gate can be made from combinations of only NAND gates or only NOR gates. These two gates represent the sigh functionality. Here two sets of inputs are given.

The NOR gate is a digital logic gate that implements logical NOR - it behaves according to the truth table to the right. A HIGH output (1) results if both the inputs to the gate are LOW (0); if one or both input is HIGH (1), a LOW output (0) results. NOR is the result of the negation of the OR operator. It can also be seen as an AND gate with all the inputs inverted. NOR is a functionally complete operation—NOR gates can be combined to generate any other logical function. By contrast, the OR operator is *monotonic* as it can only change LOW to HIGH but not vice versa. This represents the functionality of sigma. Here four inputs are given and four outputs are obtained.

In digital electronics, a NAND gate (Negated AND or NOT AND) is a logic gate which produces an output that is false only if all its inputs are true; thus its output is complement that of the AND gate. A LOW (0) output results only if both the inputs to the gate are HIGH (1); if one or both inputs are LOW (0), a HIGH (1) output results. It is made using transistors.

The NAND gate is significant because any boolean function can be implemented by using a combination of NAND gates. This property is called functional completeness. Digital systems employing certain logic circuits take advantage of NAND's functional completeness. This in combination with NOT gate represents the functionality of sigma.

In electronics, a multiplexer (or mux) is a device that selects one of several analog or digital input signals and forwards the selected input into a single line. A multiplexer of 2^n inputs has *n* select lines, which are used to select which input line to send to the output. Multiplexers are mainly used to increase the amount of data that can be sent over the network within a certain amount of time and bandwidth. A multiplexer is also called a data selector. Here a 2:1 mux is used. So combining all these components a 8 bit scalable cmos comparator is designed.

The inputs are given in set 1 namely A0-A7 and B0-B7. Set 1 compares the *N*-bit operands *A* and *B* bit-by-bit, using a single level of *N* ψ-type cells.[1] The ψ-type cells provide a termination flag D_k to cells in sets 2 and 4, indicating whether the computation should terminate.

Set 2 consists of $\sum_2$ type cells, which combine the termination flags for each of the four ψ-type cells from set 1 using NOR-logic to limit the fan-in and fan-out to a maximum of four.

The $\sum_2$ type cells either continue the comparison for bits of lesser significance if all four inputs are 0s, or terminate the comparison if a final decision can be made. Set 3

consists of $\sum_3$ type cells, which are similar to $\sum_2$ type cells, but can have more logic levels, different inputs and carry different triggering points.

A $\sum_3$ type cell provides no comparison functionality; the cell's sole purpose is to limit the fan-in and fan-out regardless of operand bit width. Set 3 provides functionality similar to set 2 using the same NOR logic to continue or terminate the bitwise comparison activity. If the comparison is terminated, set 3 signals set 4 to set the left bus and right bus bits to 0 for all bits of lower significance.[1]

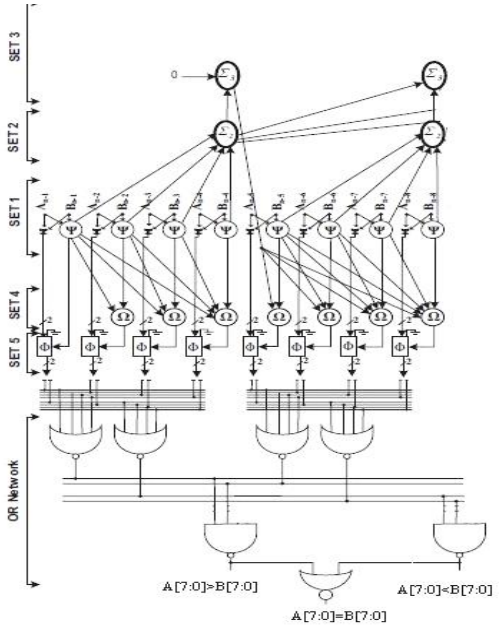

Fig 6. Modified scalable CMOS comparator

Set 4 consists of Ω type cells, whose outputs control the select inputs of Φ type cells (two-input multiplexors) in set 5, which in turn drive both the left bus and the right bus.[1]

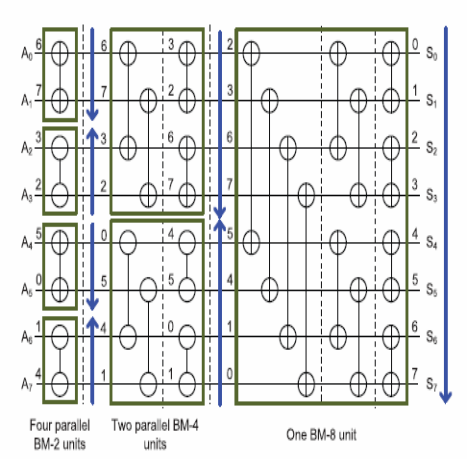

Fig 7. The CAE network for an 8-input bitonic sorting unit

For an Ω type cell and the 4-b partition to which the cell belongs, bitwise comparison outcomes from set 1 provide information about the more significant bits in the cell's Ω type cells. . Set 5 consists of *N* Φ type cells (two-input, 2-b-wide multiplexers). One input is (*Ak*, *Bk)* and the other is hardwired to "00." The select control input is based on the Ω type cell output from set 4.[1] Using this modified comparator, bitonic and odd even merge sort are constructed. The 8 input bitonic unit and odd even merge sort is shown below.

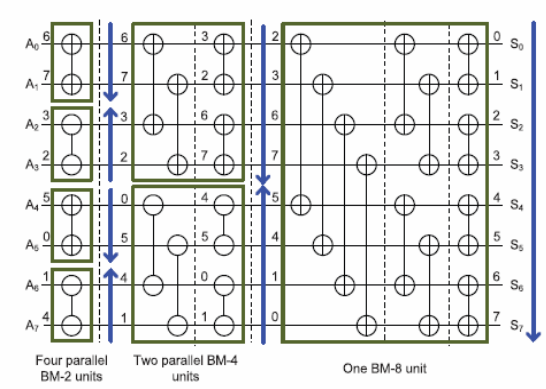

Fig 8. 8-input odd even merge sorting unit

Likewise, the bitonic and odd even max set selection and partial units are constructed.

VI. COMPARISION AND RESULTS

Here the area of bitonic and odd even partial sorting units are compared. When using the scalable CMOS comparator, the area is reduced significantly. The below fig shows the area analysis of modified bitonic and odd even circuit.

Logic Utilization	Used	Available	Utilization	Note(s)
Number of 4 input LUTs	616	4,704	13%	
Logic Distribution				
Number of occupied Slices	316	2,352	13%	
Number of Slices containing only related logic	316	316	100%	
Number of Slices containing unrelated logic	0	316	0%	
Total Number of 4 input LUTs	616	4,704	13%	
Number of bonded IOBs	128	178	71%	
Total equivalent gate count for design	3,696			
Additional JTAG gate count for IOBs	6,144			

Fig 9. area analysis of bitonic circuit

Logic Utilization	Used	Available	Utilization	Note(s)
Number of 4 input LUTs	488	4,704	10%	
Logic Distribution				
Number of occupied Slices	250	2,352	10%	
Number of Slices containing only related logic	250	250	100%	
Number of Slices containing unrelated logic	0	250	0%	
Total Number of 4 input LUTs	488	4,704	10%	
Number of bonded IOBs	128	178	71%	
Total equivalent gate count for design	2,928			
Additional JTAG gate count for IOBs	6,144			

Fig 10.Area analysis of odd even unit

The Area comparison of different sorting units is shown in the following table.

TABLE III
AREA COMPARISION OF BITONIC SORTING UNIT

	Bit by Bit comparator	Scalable CMOS comparator
No of LUT's	456	488
No of Slices	282	225
Total Equivalent gate count	3192	2928

TABLE IIIII
AREA COMPARISION OF ODD EVEN SORTING UNIT

	Bit by Bit comparator	Scalable CMOS comparator
No of LUT's	576	616
No of Slices	359	316
Total Equivalent gate count	4032	3696

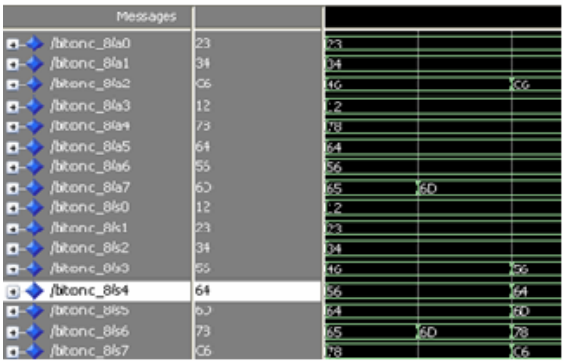

Fig 11.. Simulation results of 8 input bitonic sorting unit

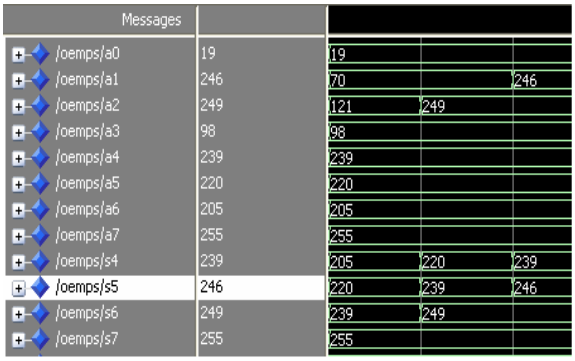

Fig 12. Simulation result of 8-to-4 odd even merge partial sorting unit

The bitonic and odd even sorting networks give the full output whereas the partial sorting unit gives the required maximum values alone. The gate count reduction is shown with the help of Xilinx tool. Compared to bitonic unit the odd even merge unit requires less gates and hence area is reduced. Odd even merge unit is better when compared to bitonic unit.

VII. CONCLUSION

The paper has presented the design and implementation of flexible, low-latency, high-throughput N-to-M sorting, and max-set-selection units and discussed the structure, performance and resource requirements of these units.[7] In this paper, we propose scalable CMOS comparator which further reduces the area. our proposed parallel designs modify the original units to obtain efficient max-set-selection and partial sorting units, reducing time and area complexities of the original algorithm . The analysis performed shows that our design have low latency. Hence this is very efficient when compared to the previous one.

References

[1] Abdel-Hafeez.S, Gordon-Ross.A,Parhami.B(2013) "Scalable Digital CMOS Comparator Using a Parallel Prefix Tree" Very Large Scale Integration (VLSI) Systems, IEEE Transcations on vol.21, no. 11, pp 1989-1998.

[2] Agrawal. J.P, Mar. (1996). "Arbitrary Size Bitonic (ASB) Sorters and Their Application in Broadband ATM Switching," Proc. IEEE Int'l Conf. Computers and Comm.

[3] Batcher.K.E, (1968) "Sorting networks and their applications" spring joint computer conference.

[4] Brajovic. V and Kanade. T, Feb. (1999), "A VLSI Sorting Image Sensor: Global Massively Parallel Intensity-to-Time Processing for Low-Latency, Adaptive Vision," IEEE Trans. Robotics and Automation, vol. 15, no. 1, pp. 67-75.

[5] Capello. G , Cola vita. A, Mumolo. E, and (1997),"A Novel Sorting Algorithm and Its Application to a Gamma-Ray Telescope Asynchronous Data Acquisition System," Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment, vol. 394, no. 3, pp. 374-380.

[6] Chakra borty. S, Cruz. R, Yun. K, James. K, Fairlie-Cuninghame. R , and ,June (2000)"A Self-Timed Real-Time Sorting Network," IEEE Trans. Very Large Scale Integration Systems, vol. 8, no. 3, pp. 356- 363.

[7] Farmahini Farahani .A, Duew.H.J, Schulte.M.J, Compton.K "Modular design of High Throughput, Low Latency Sorting Units"(2013), Computers, IEEE Transactions on vol.62, no.7,pp.1389-1402.

[8] Govindaraju. N, Gray. J, Kumar. R, and Manocha. D, (2006). "GPUTera Sort: High Performance Graphics Co-Processor Sorting for Large Database Management," Proc. Conf. Management of Data, pp. 325-336.

[9] Koch. D and Torresen. J, (2011), "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.

[10] Pitas. I and Venetsanopoulos. A.N, (1990) ,Nonlinear Digital Filters: Principles and Applications. Kluwer Academic Publishers.

[11] Pedroni. V, Jasinski. R, and Pedroni. R,(2010), "Panning Sorter: A Minimal-Size Architecture for Hardware Implementation of 2D Data Sorting Coprocessors," Proc.