Panning Sorter: A Minimal-Size Architecture for Hardware Implementation of 2D Data Sorting Coprocessors

Volnei A. Pedroni, Ricardo P. Jasinski, Ricardo U. Pedroni

LME – Laboratory of Microelectronics, UTFPR – Universidade Tecnológica Federal - Paraná
Av. 7 de Setembro, 3165, 80230-901 Curitiba PR Brazil
pedroni@utfpr.edu.br
jasinski@solvis.com.br

Abstract— **This paper describes the Panning Sorter (PanS), a new architecture for hardware implementation of compact, fast, low power data sorters operating with parallel inputs. The proposed approach is compared against several other contemporary implementations (Systolic, Bitonic, Weave, and Insertion sorters) in order to demonstrate its features. The PanS architecture is then extended to the implementation of the Minimal-Hardware Panning Sorter (MH-PanS), capable of sorting data blocks of any size with just one compare-and-swap unit, which is the absolute minimum hardware complexity for this kind of processor. Experimental results, from implementations in an FPGA device for several data block sizes, are reported.**

***Keywords*— Sorter, panning sorter, PanS, MH-PanS, FPGA.**

I. INTRODUCTION

Data sorting is a fundamental part of several data processing algorithms, like ATM switches access control [1], data distributors [2], and median plus other order statistics filters for the removal of impulse noise from images [3]. In dedicated, high-speed hardware, data sorting and sorted-data analysis can be handled by specific coprocessors, which must process a large number of inputs (N) at the same time, where N is often a power-of-two (e.g., 16, 32, 64, 128) in general processing or 9 (3×3 pixel block), 25 (5×5 block), 49 (7×7 block), or 81 (9×9 block) in image processing. In what follows, the number of bits per input will be represented by b.

This paper deals with the physical implementation of digital sorters operating as 2D (that is, with parallel inputs and parallel outputs) coprocessors. In section II, four contemporary sorter implementations, called Systolic, Bitonic, Weave, and Insertion sorter, are described. Then, in Section III, the panning sorter is described (the name "panning sorter" is to due to the panning-like movement used to order the data). In Section IV, the proposed circuit is carefully compared to all of the previous sorters. Then, in Section V, another novel architecture, the Minimal-Hardware Panning Sorter (MH-PanS), is introduced, which, to the best of our knowledge, is the most compact 2D digital sorter reported so far, derived directly from the general PanS circuit. Finally, experimental results from full implementations in FPGA devices, for several values of N and b, are presented in Section VI.

II. MAIN CURRENT SORTING CIRCUITS

A systolic sorter is shown in Fig. 1(a). It has the advantage of allowing direct pipelining, so at every clock cycle a complete sorted block is produced. Note, however, that the latency is N, and that the number of registers (each register R is b bits wide, hence b flip-flops) and of compare-and-swap (C&S) units (with two b-bit inputs) are large, that is, $N(N+1)$ and $\lfloor N/2 \rfloor (N-1)$, respectively. Details regarding the C&S circuit are shown in Fig. 1(b).

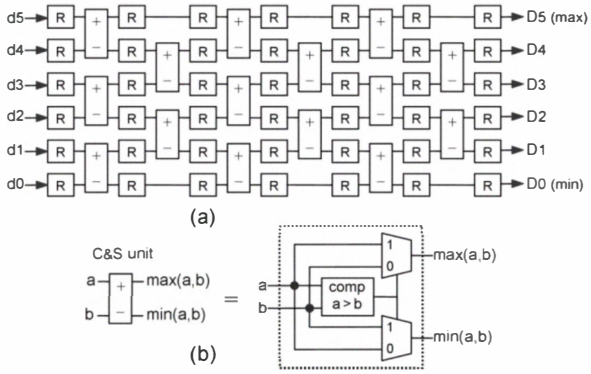

Fig. 1 a) Systolic (pipelined) sorter. The "R"s are registers, while the "+/–" "blocks are C&S units; (b) C&S circuit.

The bitonic sorter [1][2][4], also called Batcher sorter, is depicted in Fig. 2. The circuit is fully combinational, so the outputs are computed in just one iteration. Note, however, that the number of C&S layers is $0.5(1+\log_2 N)\log_2 N$, which thus represents the propagation delay through the circuit. Note also that the number of C&S units is $0.25N(1+\log_2 N)\log_2 N$, which is still a large value.

The weave sorter [5] is shown in Fig. 3. It requires less hardware than the previous two, but it is still large. The general circuit requires $2N$ b-bit registers, $3N$ $2\times b$ multiplexers (recall that a 3-input mux is equivalent to two 2-input muxes), and $N/2$ C&S units (which are more complex

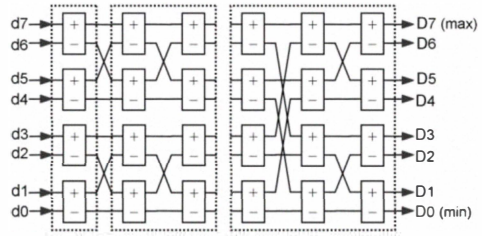

Fig. 2 Bitonic sorter (for N=8). This circuit does not scale well for non-power-of-two input sizes. The C&S units are from Fig. 1(b).

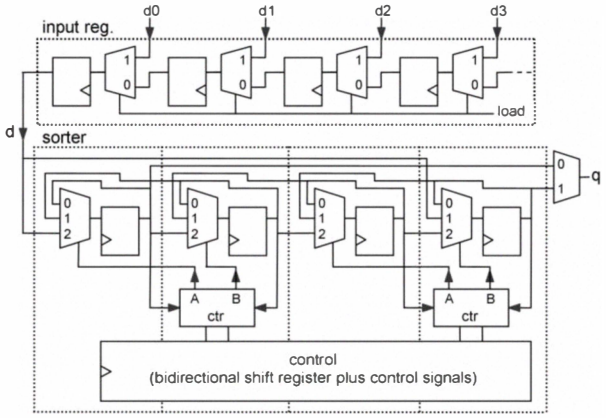

Fig. 3 Weave sorter.

than the regular C&S because they operate with control signals). The control unit (details not shown in the figure) contains N single-bit registers, N 3×1 muxes, and additional circuitry for generating the control signals. It requires N clock cycles to sort the N inputs.

Finally, the insertion sorter [6][7], operating with parallel inputs, is shown in Fig. 4. This circuit exhibits a substantial gain in terms of compactness relative to all the others. It consists of $2N$ b-bit registers, $2N$ $2\times b$ multiplexers, and N regular comparators. Again, the time complexity is N.

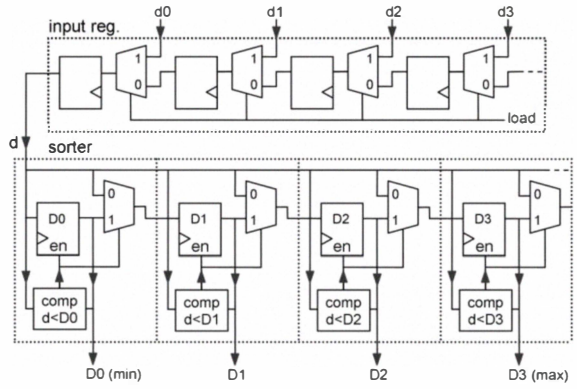

Fig. 4 Insertion sorter.

III. PANNING SORTER (PanS)

Fig. 5(a) illustrates the data-panning principle (for N=7). The circuit contains N b-bit registers (with data $x_0, x_1, \ldots, x_6$) and

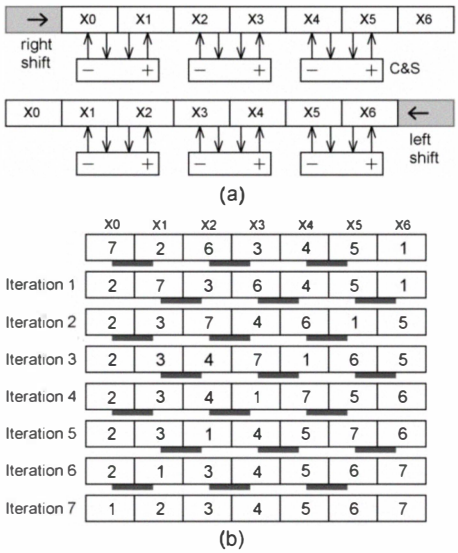

Fig. 5 (a) Data-panning principle (for N=7). The input vector is shifted to the right (1st diagram), then to the left (2nd diagram) a total of N times; (b) Numeric example (the underscore bars indicate where the C&S units are situated).

$\lfloor N/2 \rfloor$ C&S units. During the sorting procedure, the input vector is shifted one position to the right (1st diagram), then one position to the left (2nd diagram), in a panning-like movement. After a total of N shifts, the registers will automatically contain the sorted values. A numeric example is presented in Fig. 5(b), where the underscore bars indicate where the C&S units are situated.

Examples of panning sorter implementations are shown in Fig. 6, for both odd and even values of N. Initially, the inputs are loaded into the registers ($R_0 \ldots R_{N-1}$), which, after N iterations, will contain the sorted data. The left-right shifting effect is emulated by the way the C&S units are connected to the registers, plus an enable signal, eliminating the complexity of bidirectional shift registers. The enable chain is generated by a circular N-stage, single-bit shift register.

A numerical example (for N=6) is shown in Fig. 6(c), where the dark boxes indicate that a C&S unit is momentarily disabled. The moving vertical bar illustrates the shifting process; it "attracts", on the right, the smaller values, (and on the left, the larger values), so in the end the highest value will be immediately to its left.

Implementation details are depicted in Fig. 7. The circuit requires N b-bit registers, N single-bit registers, N $2\times b$ muxes, and $\lfloor N/2 \rfloor$ C&S units (with an additional enable gate). Note that in this sorter the working registers and the input registers are the same.

IV. COMPARISON

The resources usage in the PanS circuit is compared with other sorters in Tables I and II. The first shows the analytical values, while the latter shows numeric examples for b=8 bits and N=16 and 49. The compactness (which directly affects speed and power consumption) of the proposed circuit is

superior in all items, and the time complexity is only inferior to that of the systolic sorter, which, on the other hand, is the most hardware (and power) consuming.

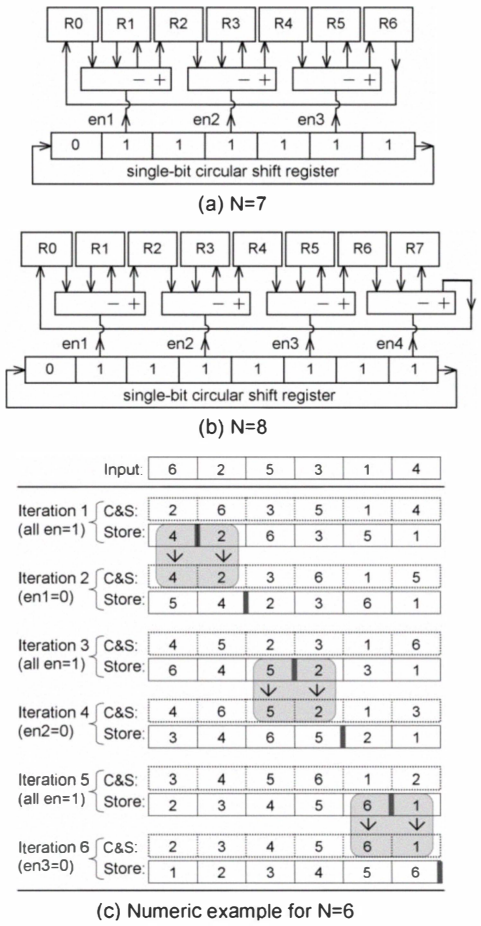

Fig. 6 Examples of panning sorter implementations for (a) N odd and (b) N even. A numerical example (for N=6) is shown in (c), where the dark boxes with arrows indicate that the C&S unit is disabled.

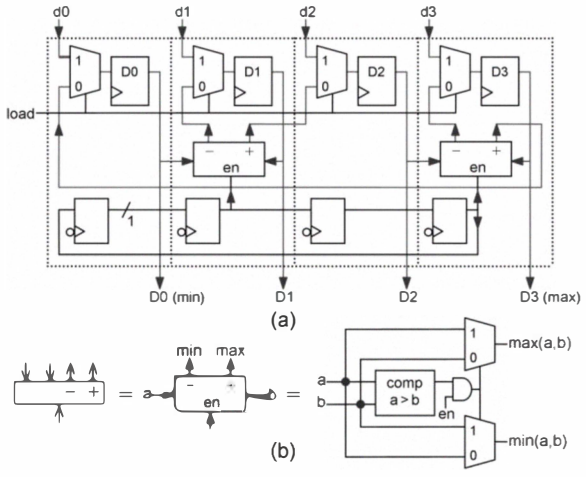

Fig. 7 Panning sorter implementation details: (a) Complete circuit; (b) C&S unit.

TABLE 1
RESOURCES USAGE (EXPRESSIONS)

Sorter type	Flip-flops	Comparators of size 2×b	Muxes of size 2×b	Muxes size 3×1	Clock cycles
Systolic	N(N+1)b	$\lfloor N/2 \rfloor (N-1)$	$2\lfloor N/2 \rfloor (N-1)$	---	1 (*)
Bitonic	---	$0.25N(1+\log_2 N)\log_2 N$	$0.5N(1+\log_2 N)\log_2 N$	---	async (**)
Weave	N(2b+1)	N/2 (***)	3N+1	N	N
Insertion	2Nb	N	2N	---	N
Panning	N(b+1)	$\lfloor N/2 \rfloor$	2N	---	N

(*) After pipeline is full (latency=N) (**) $0.5(1+\log_2 N)\log_2 N$ time delays (serial C&S units)
(***) More complex (require control signals)

TABLE 2
RESOURCES USAGE (NUMERICAL EXAMPLES)

Sorter type	N	Flip-flops	Comparat. 2×b	Muxes 2×b	Muxes 3×1	Clock cycles
Systolic	16	2,176	120	240	---	1 (*)
	49	19,600	1,152	2,304	---	1 (*)
Bitonic	16	---	80	160	---	10 (**)
	49	---	>515	>1030	---	21 (**)
Weave	16	272	8	49	16	16
	49	833	25	148	49	49
Insertion	16	256	16	32	---	16
	49	784	49	98	---	49
Panning	16	144	8	32	---	16
	49	441	24	98	---	49

(*) After pipeline is full (latency=N) (**) Time delay through C&S layers (async)

V. THE MINIMAL-HARDWARE PANNING SORTER (MH-PANS)

A new architecture, the Minimal-Hardware Panning Sorter (MH-PanS), is introduced in Fig. 8. This circuit is capable of sorting data blocks of any size with just one C&S unit, which is the absolute minimum hardware complexity for this kind of processor.

In Fig. 8(a), the general principle is presented. In Fig. 8(b), an implementation example is shown (just one C&S plus a trivial unidirectional single-bit shift register for control). A numerical example is shown in Fig. 8(c). Now the number of iterations is $N(N-1)$. Note that after every N iterations the next largest value reaches its final position (see circles in iterations 6 and 30).

The major application for this approach is in systems where speed is not a major requirement (because the time complexity is now high), but silicon space and power consumption are.

VI. EXPERIMENTAL RESULTS

Using VHDL, the two most efficient sorting architectures (insertion sorter and panning sorter) were both synthesized to a high-end Altera Stratix IV FPGA (EP3SE230F29C2) for several combinations of data width (b) × sorter length (N). The values chosen for N include lengths of interest for image processing (3×3, 5×5, 7×7, and 9×9) as well as for general data processing applications (8, 16, 32, 64, and 128). The values chosen for b are the usual 8, 16, 32, 64, and 128 bits.

The resources usage, in complete agreement with Tables I and II, is depicted in Fig. 9. It can be seen that the number of both look-up tables (ALUTs) and registers increase linearly with respect to both the configurable parameters, which is indicated by the planar shapes. It also can be seen the superior compactness of the PanS architecture, indicated by the

significantly lower logic resources usage. On average, the panning sorter requires 64% of the ALUTs and 50% of the registers compared to the insertion sorter.

As expected, the panning sorter is also slightly faster than the insertion sorter, as shown in the measurements of Fig. 10, for b=16 and b=128 bits and N ranging from 8 to 128 inputs. The speeds are comparable for small word sizes, but the difference in favor of PanS grows as the word sizes grow.

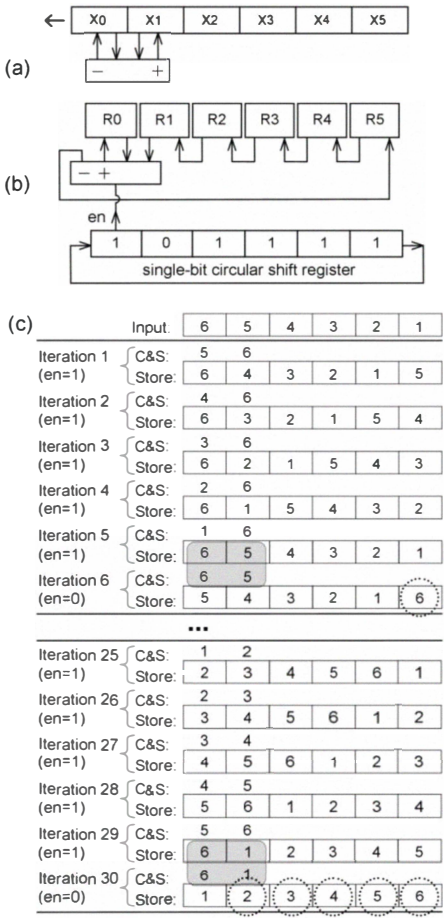

Fig. 8 Extension of the panning principle that allows the construction of a full sorter with just one C&S circuit. (a) Minimal-hardware panning sorter (MH-PanS); (b) Implementation example; (c) Numeric example.

VII. CONCLUSIONS

We have described the panning sorter (PanS), a new architecture for hardware implementation of digital data sorters. A minimal-hardware sorter (MH-PanS), derived from the original architecture, was also introduced, which is proper for systems where speed is not a major requirement (because its time complexity is now $N(N-1)$), but silicon space and power consumption are. The overall strengths of the main competitors, particularly the very efficient insertion sorter architecture, were also highlighted and evaluated.

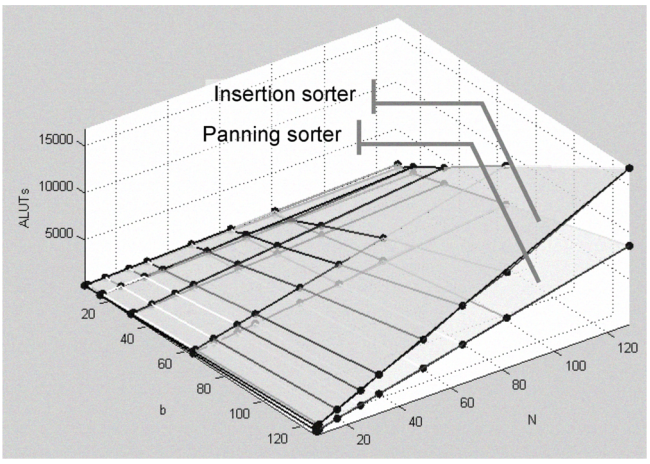

Fig. 9 Logic resources usage (look-up tables) for the presented sorter architectures, as a function of N (number of words) and b (bits per word).

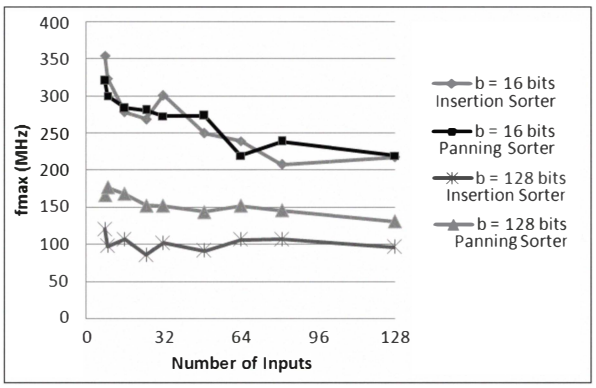

Fig. 10: FPGA implementation maximum speeds for the insertion and panning sorters as a function of N for b=16 and b=128 bits.

REFERENCES

[1] G. Agrawal, "Arbitrary size bitonic sorters and their applications in broadband ATM switching," Int. Phoenix Conference on Computers and Communications, 1996.

[2] J. G. Lee, B. G. Lee, "Realization of large-scale distributors based on Batched sorters," IEEE Trans. on Communications, vol. 47, no. 7, July 1999.

[3] K. Aiswarya, V. Jayaraj, D. Ebenezer, "A new and efficient algorithm for the removal of high-density slat and pepper noise in images and video," Int. Conference on Computer Modeling and Simulation, 2010.

[4] T. H. Lee, J. J. Chou, "Some topological properties of bitonic sorters," IEEE Trans. on Computers, vol. 47, no. 9, Sept. 1998.

[5] A. Mukherjee, N. Motgi, J. Becker, A. Friebe, C. Habermann, M. Glesner, "Prototyping of efficient hardware algorithms for data compression in future communication systems," Int. Workshop on Rapid System Prototyping (IWRSP'01), 2001.

[6] L. Ribas, D. Castells, J. Carrabina, "A linear sorter core based on programmable register file," Conference on Design of Circuits and Integrated Systems (DCIS'04), 2004.

[7] S. Dong, X. Wang, "A novel high-speed parallel scheme for data sorting algorithm based on FPGA," Int. Congress on Image and Signal Processing (CISP'09), 2009.