FPGA IMPLEMENTATIONS OF SORTERS FOR NON-LINEAR FILTERS

Boaz Hirschl and Leonid P. Yaroslavsky

Tel-Aviv University, Faculty of Engineering, Dept. of Interdisciplinary Studies, Tel Aviv 69978, Israel

phone: +972 09 8942534, fax: +972 9 9611441, email: Email: boaz.hirschl@intel.com

ABSTRACT

Nonlinear filters for image processing have attracted considerable interest due to their ability to preserve edges, denoise images and to their potential for image enhancement and segmentation. One of the basic operations in nonlinear filtering is sorting data from a 2-D sliding window. In this paper we propose a novel approach to hardware FPGA implementation of data sorting, suggest three implementations and compare them.

1. INTRODUCTION

Nonlinear filters constitute a wide family of filters with proven capabilities in image denoising, enhancement and segmentation[1]-[3]. The filters work in a sliding window and generate an estimate of the window central pixel by applying certain estimation operations to a subset of the window pixels. This subset is referred to as the central pixel's neighbourhood [2].

One of the fundamental operations involved in non-liner filtering is data sorting. Numerous methods have been suggested for both software and hardware implementation of the sorting operation. The suggested architectures can be divided into two types: parallel and serial. Parallel sorters receive, in a parallel form, a vector of data and produce, after some processing time, the sorted output vector. These sorters are suitable for "off line" processing of the images, which are stored in memory. In [4] a very large scale integration (VLSI) column sort architecture in which a sorter based on matrix of inputs and outputs with selection switching points is implemented to sort columns. This solution is compact, fast but supports a limited number of items.

In our approach to parallel processing, the rank of each number is calculated [2]. This results in a rank vector and enables easy extraction of any order statistics such as Min, Max, and Median values. To implement this approach, two stages of logic calculate each number's rank. The first stage is either comparator or adder and the second stage is adder. The architecture is compact in terms of silicon resources.

Serial sorters receive the data, one number or bit after the other. The serial sorters are well suited for real-time image capturing and transmission when the processor is a part of the data stream. In [5], a serial sorting architecture of odd/even transpose is suggested. In [6] a serial rank calculation is performed in order to convert input value to its rank. These sorters are recursive by their nature and each output vector is computed using previously computed vectors. A hybrid approach uses both serial and parallel architecture. A hybrid sorter combining a serial software sorter and parallel hardware architectures is evaluated in [7].

The architectures discussed in the paper, sorters of N elements vector where $N = n * n$ and n is the window size. In an actual application a pre-processor will arrange the two-dimensional window into the input vector. The size of the vector can vary in a wide range limited only by silicon area.

Serial sorters use a First In First Out (FIFO) buffers. Two FIFO algorithms are discussed. Serial FIFO sorter uses the magnitude attribute to insert each new pixel into its correct place. On the same cycle, during the complementing phase, the oldest pixel leaves the sorted array. A second implementation is a serial rank computer where each pixel that enters the array causes all the other pixel rank attributes to be updated. In a similar fashion, the pixel, which is leaving, causes all the pixel rank attributes to be re-adjusted. This is an improvement of the system suggested in [6].

In the work, VHDL sorter generator software was developed and used to generate the sorter's VHDL files with variable window size. The WebPACK ISE 5.1.03i by Xilinx Software Solutions was used for synthesis and implementation. The ModelSim SE/EE PLUS 5.4e by Model Technology along with Matlab by Mathworks was used for simulation and verification.

2. PARALLEL RANK COMPUTER (PRC)

The rank computer receives, in a parallel fashion, a vector of N numbers and produces their ranks in two clock cycles. By comparing every pair of numbers and summing the comparison values the rank of each number is calculated. Every pair of numbers is computed and the result is hard limited to ± 1 using a look-up table (LUT). All these results difference between, are accumulated in accordance with the index of the rank computed as is illustrated in Figure 1.

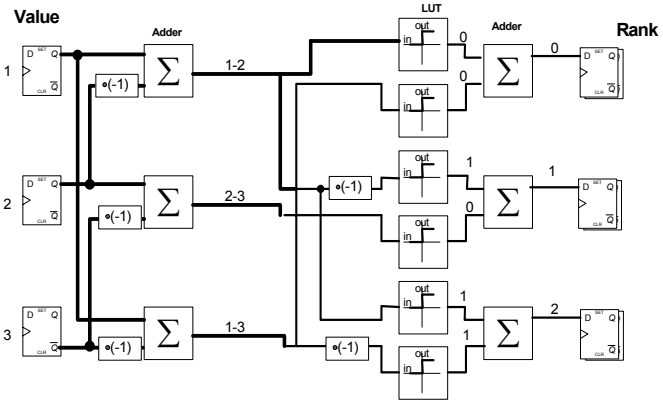

Figure 1: A Rank Computer for Three Numbers

For the first element of the vector of elements, all the pairs that include the index '1' subtractions ('1-2','1 -3') values are hard limited and accumulated giving the result of the elements rank.

Hardware optimizing is shown in Figure 2 where, instead of adders, comparators are used. This implementation utilizes specific comparator cells that reside in the FPGA and their one-bit outputs.

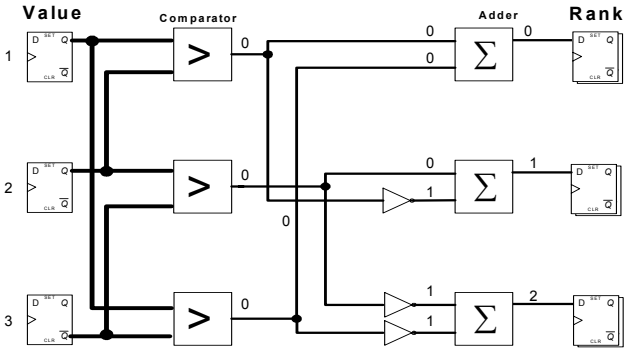

Figure 2: Optimized Rank Computer

In the second stage of the sorter, the adder is a simple adder that sums the comparators results and outputs the rank.

The adder size depends on the number of elements N while latency is always two clock cycles.

3. SERIAL FIFO SORTER (SFS)

A First In First Out (FIFO) store input vector data in the order that it is received. The Serial FIFO Sorter (SFS) is different from regular FIFO in a way that the input vector data is ordered by its magnitude, still this data leave the sorter in a FIFO way. At each clock cycle, one number enters its place in the sorter according to its value; on the same clock cycle, one number leaves the sorter. In Figure 3 b), when the number 5 enters the sorter from the left in the first phase of the clock, a shift to the right takes place. In Figure 3 c) in the second phase of the clock, when '0' leaves the FIFO, the numbers are shifted back to the left.

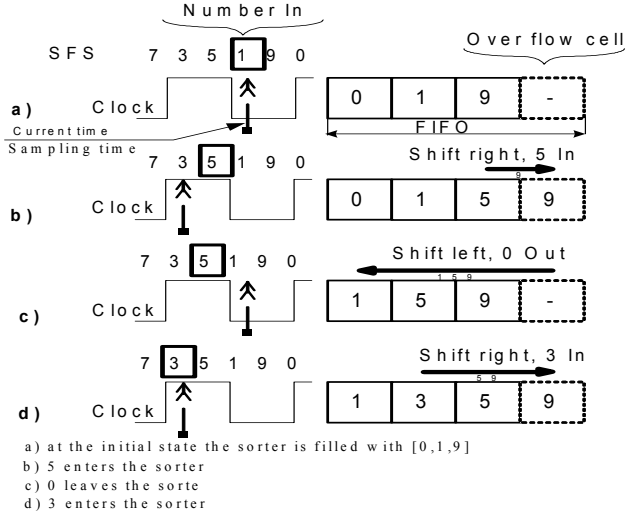

Figure 3: An example of a Serial FIFO Sorter

The sorter consists of two levels of memory elements to enable it to operate in one clock cycle. These two memory arrays, which work on the first and second phases of the clock respectively, are labeled "sorted vector" and "shadow sorted vector". Each memory element stores both the value and the "age" of the number. The term "age" describes the order in which a number entered the FIFO. When a number enter the FIFO its age is one, each clock cycle the age is increased by one. When the age value is equal to the FIFO size the number should leave the FIFO and this value equal to the FIFO size is referred to as "retirement age". The FIFO first out action is determined by the age attribute.

The implementation of this sorter is based on a cell that includes memory elements: main and shadow ones and some logic elements to support the sort and shift operations. A schematic diagram of the sorter basic cell is shown in Figure 4.The upper part is the main sort vector cell and the lower part is the shadow vector cell. In the first phase of the clock, the upper logic selects which value to transfer (A_out) to the lower part: the cell value, the new number in (B_in) or the number to the left of the cell (An-1, a shift right operation). The main selector behavior is described by the formula:

$$A\_Out = \begin{cases} A_n & B > A_n \\ A_{n-1} & B < A_n, B > A_{n-1} \\ B & B > A_n, B > A_{n-1} \end{cases}$$

In the second phase, the lower logic selects which number (C_out) to transfer up, either the number in the cell or the number to the right (a shift left operation). The decision is based on the following rules: if the cell age reached the retirement age then the number to the right (C_{n+1}) is shifted out. If a number was retired then again, the number to the right is shifted out; otherwise the current number is shifted out. The shadow selector behaves is described by the formula:

$$C\_Out = \begin{cases} C_n & Age_n < Age_{retirement} \,\&\, reteire_{n-1} = false \\ C_{n+1} & Age_n = Age_{retirement} \\ C_{n+1} & reteire_{n-1} = true \end{cases}$$

The cells allow shift in both directions.

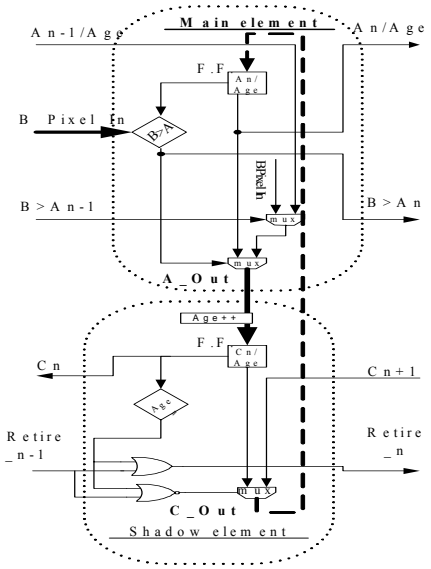

Figure 4: Serial FIFO Sorter Basic Cells

4. SERIAL RANK COMPUTER (SRC)

A serial rank computer (SRC) operates on a FIFO basis that includes two attributes: value and rank. The algorithm of the implementation is based on [6]. The SRC FIFO numbers are arranged according to their arrival sequence accompanying each number with its calculated rank, this is unlike SFS that order the numbers in the memory array according to their value. In Figure 5 we see the basic memory element that stores value and rank. In this example, the FIFO serial rank operator starts with ['1','9','0'] and the respective ranks ['1','2','0']. As the numbers enter the array and leave the array of the sorter, the ranks are updated every clock cycle.

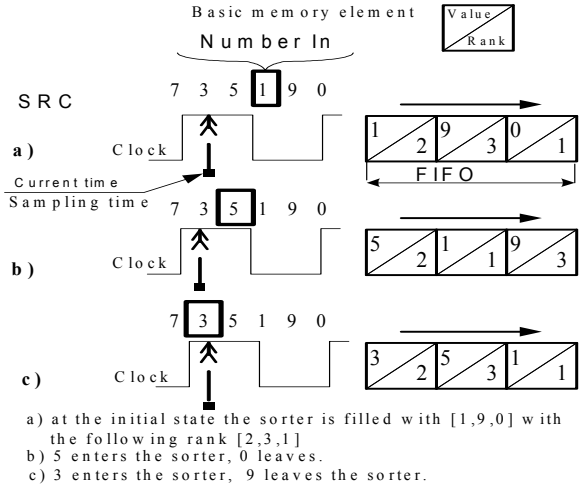

Figure 5: Example of 3 Number Rank Sorter

A group of rank cells forms the rank computer. The first rank cell calculates its rank by summing the results of the comparison of the new value to all the values in the sorter. The other rank cells compare their value to the new number and the oldest number that leaves the FIFO and adjust their rank accordingly.

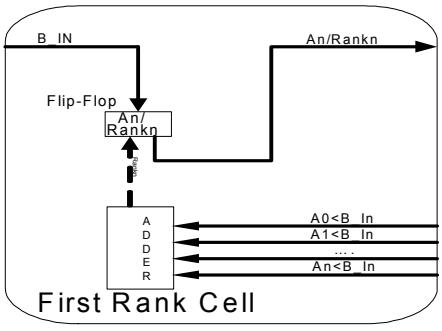

Figure 6: First Rank Cell

The first rank is created by summation of all the comparisons.

The other rank cells calculate their rank as follows:

$$Rank_n = \begin{cases} Rank_n = Rank_n - 1 & Rank_n >= Rank_{Last} \\ Rank_n & Rank_n < Rank_{Last}, B_{in} < A_n \\ Rank_n = Rank_n + 1 & B_{in} < A_n \end{cases}$$

This implementation includes (B_in) and the last cell rank (last_rank) inputs which are compared with the current cell and are used to control the adder and subtract shown in Figure 7.

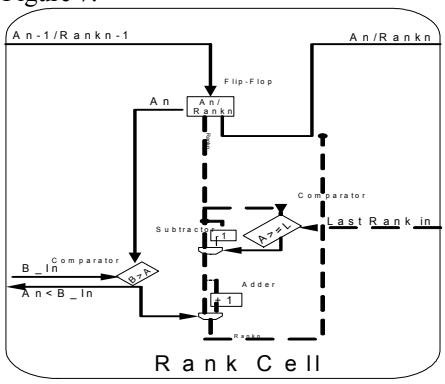

Figure 7: Rank Cell

5. EXPERIMENTAL RESULTS AND ANALYSIS

This section describes the results of the different experiments with the three architectures on Xilinx VirtexE devices for different window sizes. The following acronyms will be used.

SFS – Serial FIFO Sorter

PRC – Parallel Rank Computer

SRC – Serial Rank Computer

The benchmarks we examine are performance, speed, area, power and latency. The speed is measured in MHz and is dependent on the maximum delay between two state elements or one state element and Input/Output pin. Some of the speed results are normalized to the slowest implementation. The area is measured in FF numbers, LUT numbers and equivalent number of gates. The power is a function of area, frequency and activity factor. The latency is dependent on the architecture of the algorithm and can be constant for an algorithm or vary with different window sizes.

- Speed Analysis

The sorter's speed varies from 100Mhz to 200Mhz. For reference, an 8-bit counter that creates the data-sorted valid signal works at 300Mhz. The sorting speed is not a function of the window size. The speed comparison is presented in Table 1: for windows of up to 7x7.

Algorithm	Speed
Serial FIFO Sorter (SFS)	1.2
Parallel Rank Computer (PRC)	1.66
Serial Rank Computer (SRC)	1

Table 1. Algorithm Speed Comparison. Speed unit is 96Mhz

The parallel algorithms are the fastest. The serial sorter is designed to sort in one clock cycle using both phases of the clock but the complexity of its cells causes more delays. The SRC is the most complex and therefore has the slowest architecture.

- Area Analysis

The parameters that are obtained in the experiments are the number of FF, LUT and of gate equivalents. Table 2 shows the area for a 5x5 filter. We see that the SRC suggested in this paper is the most compact algorithm. The serial algo-

rithms, both the SFS and the SRC, are quite compact. The PRC is compact and can compete with the serial algorithms.

Table 2: Algorithms Area for 5x5 Window

	FF	LUT	Gates
SFS	700	1875	18362
PRC	450	3270	31547
SRC	408	1302	14488

Table 3 shows the required number of FF, LUT and gates per one window number.

Table 3: Algorithms Area Comparison

	FF	LUT	Gates
SFS	$O(28N)$	$O(70N)$	$O(710N)$
PRC	$O\left(\left\lfloor\frac{N^2}{2}\right\rfloor + N\lceil LOG_2 N\rceil\right)$	$O\left(8N\left\lfloor\frac{N}{2}\right\rfloor + N^2\right)$	$O(160N^2)$
SRC	$O(16N)$	$O(52N)$	$O(582N)$

The SFS consumes 28 FF per number (8 bits for value and 6 bits for age (for $N \le 64$) in two cells - main and shadow ones, 8 x 2 + 6 x 2 = 28). The PRC consumes $\left\lfloor\frac{N}{2}\right\rfloor + \lceil \log_2(N)\rceil$ FF for each number. ($\left\lfloor\frac{N}{2}\right\rfloor$ for the number of pair comparators and $\lceil \log_2(N)\rceil$ for the rank FF at the rank adders). The SRC consumes 8 FFs for rank and 8 FFs for value. The silicon area required by the algorithms is dominated by two issues, logic and routing resources. The logic includes the LUTs and the FFs. Routing includes the internal metal connection between different logic elements and the pins connecting the FPGA to the outer world.

For FIFO-type algorithms, the pin number and routing resources are minimal and the bottleneck is the number of FFs and LUTs.

In terms of FFs and LUTs FIFO-type algorithms linearly depends on the size of the vector processed. The parallel sorter number of FFs and LUTs increases in square ratio.

- Latency Analysis

As a part of the validation suite a valid signal is created. The valid time is equal to the latency. Each algorithm implementation was accompanied with a counter that created this valid signal to enable verification. The counter value was calculated in the generation script and hard coded into the VHDL files.

The serial algorithms latency is proportional to the number of stages in the FIFO. The parallel rank computer has a fixed latency of two clock cycles.

- Power Analysis

The power consumed by the FPGA is divided into static and dynamic power consumption. Static power consummation is caused by a current leakage on the devices and can be found in the data sheet for the VIRTEX family. Switching of transistors causes dynamic power consumption. Activity factor (AF) is a parameter that describes what percentage of the device is working in an average clock cycle. The algorithm's AF in most cases is 1 since the whole device is working all the time. The serial FIFO sorter works twice in each clock cycle but in each phase of the clock only half of the cell functions there fore the SFS AF is 2.

6. CONCLUSIONS

Three modifications of FPGA implementations of sorters for non-linear filters are suggested in this paper:

- Parallel Rank Computer
- Serial FIFO Sorter
- Serial FIFO Rank Computer.

The implementations are compared in terms of:
speed, area, latency and power.

It follows from the analysis that the speed differences are not high and the main issue is the FPGA device speed.

The serial rank computer is the slowest and the simplest. The serial sorter is the middle in terms of size and speed. The parallel rank computer is the fastest implementation. The latency and size are linearly dependent on the window size for the serial algorithms. The size is $\frac{N^2}{2} + N\log(N)$ proportional to the window size in parallel algorithms. The parallel rank computer produces results in a small, fixed number of clock cycles. The power consumption is mainly a function of size and the parallel rank computer is the worst in this parameter. Advanced FPGA can sort parallel a 7 x 7 window at 200 MHz, which is about 10 times faster than the 2Ghz Pentium CPU. The work offer building blocks for non-liner filter set.

REFERENCES

[1] J. Astola, P. Kuosmanen, Fundamentals of Nonlinear Digital Processing, CRC Press, Boca Raton, N.Y., 1997

[2] L. Yaroslavsky, Nonlinear Filters for Image Processing in Neuromorphic Parallel Networks, Optical Memory and Neural Networks, vol. 12, No. 1, 2003

[3] L. Yaroslavsky, Digital Holography and Digital Image Processing, Kluwer scientific publications, Boston, 2003, ch.12.

[4] R. Lin, S.Olariu, "Efficient VLSI Architecture for column sort". IEEE Transactions on VLSI system Vol 7, NO 1, March 1999.

[5] C. Hennind, T. G. Noll, "Architecture And Implementation Of BitSerial Sorter For Weighted Median Filter". Custom Integrated Circuits Conference, Proceedings of the IEEE 1998, pg 189–192, University Of Technology RWTH Aachen, Germany.

[6] L.Lin, G.B. Adams II, E.J. Coyle, "Input Compression and Efficient Algorithms and Architectures for Stack filters". IEEE proc. Winter Workshop on non linear digital signal processing, Tempere Finland pp.5.2-5 Jan 1993

[7] M. Bednara, O. Beyer, J. Teich, R. Wanka, "Tradeoff Analysis And Architecture Design Of Hybrid Hardware/Software Sorter", Application-Specific Systems, Architectures, and Processors, 2000. Proceedings., 10-12 July 2000 pg 299 – 308.