Parallel Numerics '05, 151-163 M. Vajteršic, R. Trobec, P. Zinterhof, A. Uhl (Eds.)
Chapter 6: Multimedia ISBN 961-6303-67-8

Parallel Image Processing on Configurable Computing Architecture

Andriy Lutsyk[1,*], Oleksiy Lutsyk[2], Olexandr Pelenskyy[3]

[1] Institute of Physics and Mechanics of the Ukrainian National Academy of Sciences, Naukova 5, 79601 Lviv, Ukraine

[2] Ivan Franko National University of Lviv, Kyrylo and Methodyi 8, 79005 Lviv, Ukraine,
[3] Westukrainian College, Chuprynka 130, Lviv, Ukraine

The recent advances in imaging technology make popular using images in different branches of human activity. In the paper, a reconfigurable computing architecture based on the homogeneous computing structure concept with possibility of hardware modification in response to changes in processing environment and tuning on implementation of image filtering and segmentation algorithms in real time is presented.

1 Introduction

The recent advances in imaging technology make popular using images in different branches of human activity. Robotics, biomedical applications, industrial process control and environmental control are among them. Each procedure in this environment demands variety of processes, methods and hardware. To receive satisfactory results it is desirable to have both sophisticated methods and algorithms for image processing and compression which require considerable computational cost, and fulfilment this task in real time or near to real time.

Nowadays various architectures are suggested for highly efficient image processing, including parallel processors of the SIMD type, multiprocessor systems, pipelined processors, systolic arrays and pyramid machines. In the

*Corresponding author. E-mail: lutsyk@ah.ipm.lviv.ua

last years configurable computing devices are developed which may be able to adapt their hardware almost continuously in response to changes in the input data or processing environment [1, 2]. The configurable computing device is based on field-programmable gate arrays (FPGAs) - tuned hardware circuits that can be modified during use.

This article is organised so that first the brief description of configurable architecture based on homogeneous computing structure is presented in Sec. 2. In Section 3, a microprogram module for address finding of the maximal or minimal number from two numbers is described in detail. Section 4 is dedicated to the implementation of trimmed mean filtering on the homogeneous computing structure. Mapping of a structure-adaptive image filter on the homogeneous structure is described in Sec. 5. Concluding remarks are given in Sec. 6.

2 Configurable architecture based on homogeneous computing structure

The concept of the homogeneous computing structures was originated in the Former Soviet Union [3]. The homogeneous computing structure (HCS) can be modified according to needs of data execution and can be used for speeding up of the majority of image processing and visual data compression algorithms which are applied in control, biomedicine and multimedia [4, 5]. In this paper, the configurable computing architecture based on the homogeneous computing structure is proposed. The architecture consists of two types of matrices: the matrix of computing cells and the matrix of storage cells. The processor cell performs computations and has a transit communication channel to transmit the information without changes. The storage cell is used to store the intermediate computation results. The processor cell can execute following simple operations: addition of two one bit numbers; logical multiplication; logical multiplication with inversion; modulo 2 addition; memorizing "1"; microconstant generation.

The process of homogeneous computing structure tuning and the process of computation are shared in a time. In the beginning, a stream of instructions is formed for HCS tuning and is written to the instruction registers of every computing and storage cell, and after that the stream of data is entered. During data execution the instruction codes are kept in the instruction registers of computing and storage cells. Entry of the instruction codes is carried out through the chains of instruction code ports using the control signal. When computational requirements are changed, the new stream of instruction codes is entered to the structure swapping hardware configuration of the HCS. Thus,

it is possible to execute a series of tasks in rapid succession.

One clock generator synchronizes the work of all cells of HCS and HSS. The information streams from the input device applied to the information inputs of the computing and storage matrices are processed in accordance with the instruction codes, moving synchronously with the clock cycles from one cell to another in the matrices of computing and storage structures. The system has not a control block, a main memory and a data bus. The main peculiarity of the organization of computing process on the HCS is the conformity of specialized processor or, in the other words, microprogram module to each program instruction written in the program language of homogeneous computing structure. Many data streams are processed simultaneously, both in parallel and pipeline modes. Data streams may split, merge, exchange or cross information. This is called multipipelined execution. It is, therefore, important to represent signal and image processing algorithms in multipipelined form to be properly mapped onto the homogeneous computing structure

The homogeneous computing structure is tuned on a given data processing algorithm by storing of an appropriate sixteen bits code to the program register of the computing cell, which is an ordinary shift register. To speed up the tuning process, it is necessary both to rise the frequency of clock pulses and to increase the number of gates for the entry of program codes.

3 Construction of a microprogram module for address finding

For the implementation of a microprogram module for address finding of the maximal or minimal number from two numbers, the module max-min*(L,M)* operation can be used. Consider this module. The algorithm for design of the module can be represented as follows:

1. The determination of the difference *V=L-M*;

2. The application of the operation of "memorizing of 1", the sign of the number V is memorized, that is, a value is formed

$$Q = \begin{cases} 11..11 \;\; if \;\; Z = 1 \;\; (M > L) \\ 00..00 \;\; if \;\; Z = 0 \;\; (L \geq M) \end{cases}$$

   where Z is a value of the elder sign bit (the *n-1* bit in modified complimentary code)

3. The formation $P = \overline{Q} \wedge L$, $R = Q \wedge M$;

4. The formation $P' = Q \wedge L$, $R' = \overline{Q} \wedge M$;
5. $max(L, M) = P \oplus R$;
6. $min(L, M) = P' \oplus R'$;

   Only four steps are added in the algorithm for the construction of the microprogram module for address finding of the maximal and minimal numbers from two numbers.

7. The formation $D = \overline{Q} \wedge H$, $F = Q \wedge G$;
8. The formation $D' = Q \wedge H$, $F' = \overline{Q} \wedge G$;

   Here H and G are addresses of the numbers L and M respectively.

9. $(H, G)_{max(L,M)} = D \oplus F$;
10. $(H, G)_{min(L,M)} = D' \oplus F'$;

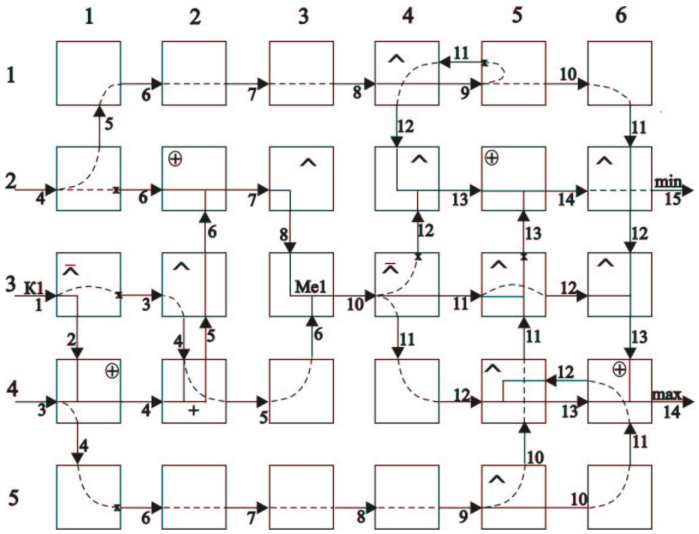

Figure 1: The microprogram module of max-min*(L,M)* operation

The task of the microprogram module max-min*(L,M)* is the determination of a larger and smaller number from two numbers and passing the larger number to one output and the smaller number to another output (Fig.1). The microprogram module of max-min*(L-M)* can be used in median filtering of images or mean trimmed filtering of images, where is applied as an element for parallel-pipeline exchange sorting. Instead of, the microprogram module

of finding address of the maximal and minimal number from two numbers, additionally to the function of finding the larger and smaller number and directing them to appropriate addresses, links larger and smaller numbers with its addresses, which they had on the inputs of the module, and directs these addresses together with the numbers to appropriate outputs. For example, assume that the larger number entered the module input with the address 2, and the smaller number with the address 1. Then on the output of the module, which is determined for the output of the larger number, the address 2 of the larger number will appear together with the larger number, and on the output, which is determined for the smaller number, the smaller number will appear together with its address 1. Such module, using the scheme of exchange sorting can be used, for example, for finding an area address with minimal value of absolute or standard deviation which corresponds to the most homogeneous region of a window.

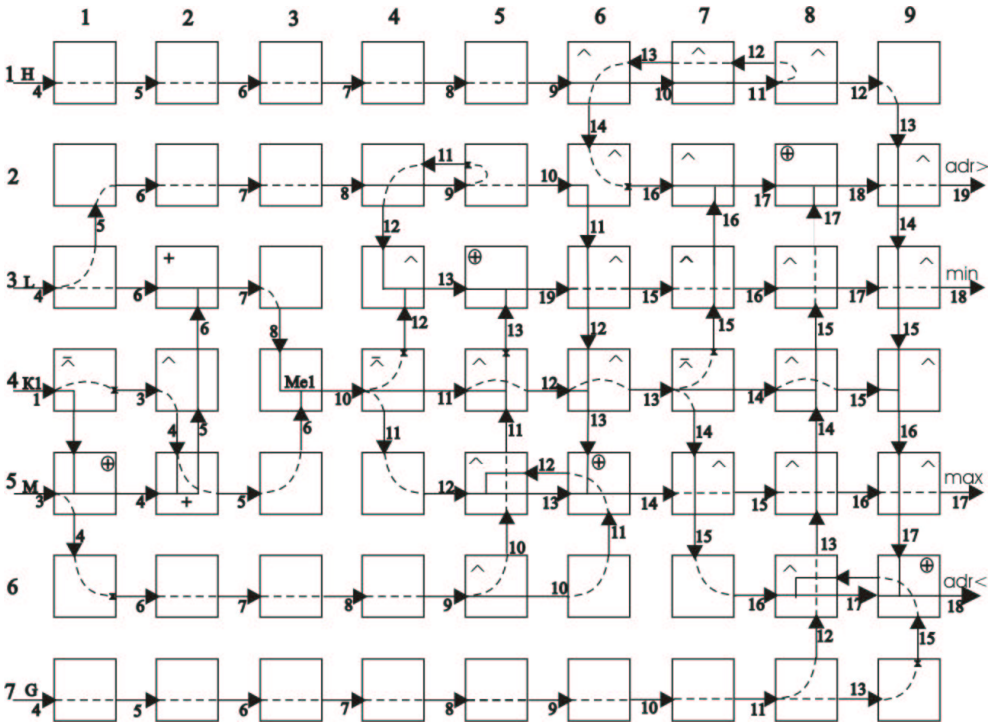

Figure 2: Microprogram module of finding address of the maximal and minimal number from two numbers

For effective processing of numbers in the homogeneous structure which enter the module in bitwise sequence, a modified supplementary code is used. The modified supplementary code is as follows: n, n-1, n-2, n-3, ..., 2, 1, where

n is a buffer bit, n-1, n-2 are sign bits (n-1 is the first sign bit), and n-3, ..., 2, 1 are bits of a mantissa. Two numbers and the constant K_1 enter the module inputs in bit-by-bit sequences starting from the low-order bits and the result appears at the output also starting from low-order bits. Bitwise numbers enter the module and are processed there one by one without any intervals.

Consider the work of the microprogram module of finding address of maximal or minimal number (Fig.2). The size of this module is 7×9 cells. It exploits 33 computing cells more compared to the module max-min*(L,M)*.

According to the algorithm written above, the determination of the difference V=L-M of two numbers is carried out in the first step. This function is performed by a group of the cells (3,1), (3,2), (4,1), (4,2), (5,1), (5,2). The cells (4,1), (5,1) and (5,2) handle the operation of sign change of the number M. The algorithm of the operation of sign change is in its turn divided into three steps.

- Inverting of the constant K_1 (K_1=00...001 and has the same length as all processed numbers);
- Modulo 2 addition of the number M and the inverted constant K_1, displaced on the one cycle ahead;
- Addition of the result of the previous step and the constant K_1.

The cell (3,1) inverts the constant K_1. Modulo 2 addition is handled in the cell (5,1) and the result of this operation is added to the constant K_1 in the cell (5,2). The operation of addition of numbers L and *(-M)* with taking into account carry bit, is performed in the cell (3,2).

The cell (4,3) carries out the next step of "memorizing" sign of the number V. The number V enters the input 2 of the cell (4,3) in the eighth cycle and the constant K_1 enters the input 4 in the sixth cycle. Since, the constant K_1 outruns the number V, then the value of high sign bit of the number V coincides with bit of the constant K_1 which has value 1. In Table 1, the number sequence, which enters the input 2 of the cell (4,3), and a sequence of the constant K_1 on the input 4 of the same cell are presented. Two numbers $0ZZa_{n-3}..a_1$ and $0Z^{'}Z^{'}a^{'}_{n-3}..a^{'}_1$ enter consecutively the input 2. The constant, which is displaced in two cycles ahead, enters the input 4. That is, constant bit to have value 1 enters the cell simultaneously with the first sign bit Z of the first number in the eighth cycle and with the first sign bit $Z^{'}$ of the second number in the (n+8)th cycle. Therefore, the value of the first sign bit Q will appear in the output of the cell (4,3) after 2 cycles, and it will be kept by the cell for n cycles to the moment of arrival of the first sign bit of the second number and respectively of bit with value 1 of the constant K_1.

Table 1. The number V enters the input 2 of the cell (4,3) in the eighth cycle and the constant K_1 enters the input 4 on the sixth cycle. Since, the constant K_1 outruns the number V, then the value of high sign bit of the number V coincides with bit of the constant K_1 which has value 1.

Cycle	*7*	*8*	*9*	*10*	...	*n+6*	*n+7*	*n+8*	*n+9*	*n+10*	...	*2n+6*
Input 2	*0*	*Z*	*Z*	a_{n-3}	...	a_1	*0*	$Z^{'}$	$Z^{'}$	$a^{'}_{n-3}$	...	$a^{'}_1$
Input 4	*0*	*1*	*0*	*0*	...	*0*	*0*	*1*	*0*	*0*	...	*0*

The third step of formation $P = \overline{Q} \wedge L$, $R = Q \wedge M$ is carried out in the cells (4,6) and (4,5), respectively. $P' = Q \wedge L$ and $R' = \overline{Q} \wedge M$ are formed in the cells (3,4) and (4,5) in the fourth cycle to be tuned on the operation "logical multiplication".

The maximal value of two numbers $max(L, M) = P \oplus R$ is determined in the cell (5,6) using modulo 2 addition of values P and R formed previously. The same operation of finding minimal value of two numbers $min(L, M) = P' \oplus R'$ is performed in the cell (3,5).

In the seventh step, values $D = \overline{Q} \wedge H$ and $F = Q \wedge G$ are formed in the cells (2,7) and (4,8) for further finding an address of the maximal number. The cells (4,9) and (6,8) are used for the formation of values $D' = Q \wedge H$, $F' = \overline{Q} \wedge G$.

And, at last, the address of the larger number $(H, G)_{max(L,M)} = D \oplus F$ is determined in the cell (2,8) and the address of the smaller number $(H, G)_{min(L,M)} = D' \oplus F'$ is determined in the cell (6,9).

As it can be seen from Fig. 2, the larger number leaves the module from the output of the cell (5,9) in the seventeenth cycle, and the smaller number leaves the module from the output of the cell (2,9) in the nineteenth cycle. The address of the larger number appears in the output of the module from the output of the cell (2,9) and the address of the smaller number will be in the output of the module in the eighteenth cycle (cell (6,9)).

4 Implementation of trimmed mean filter

It is known that the mean filtering is better for suppression of normal white noise in image and the median filter is better for the elimination of impulsive noise. The median filters preserve sharp edges well, but they may course some distortion to corner edges. Trimmed mean filter combines better properties of these two filters and can be used for image filtering with a mixed conditional distribution. Further generalization can be achieved using so-called (α, β)-trimmed mean filtering [6], where the parameter α determines the percentage

of the retained pixels and the parameter β is the shift of the subset of sorted pixels to be averaged. The inner trimmed mean filter is more robust against outliers (impulsive noise) and the outer trimmed mean filter is better in the case of uniform noise distribution. The outer trimmed mean filter rejects central pixels in the ordered sequence and averages remaining pixels of this sequence.

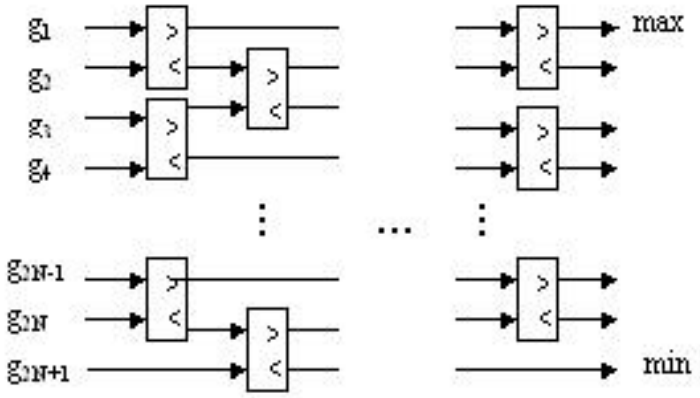

Figure 3: Implementation of parallel-pipeline sorting of *2N+1* numbers on the homogeneous computing structure

Consider the implementation of the trimmed mean image filter on the homogeneous computing structure. The homogeneous storage structure is used for the formation of running window *W(i,j)*. Pixels are delayed for several lines according to the size of a window (for the window 3x3, for instance, the delay is carried out on one and two lines). In the next stage, all window elements are ordered. Microprogram modules of operation to find the maximum of two numbers (min/max) are used for fulfilment of the ordering operation. This module performs sixteen-bits numbers. The module has two outputs: the first one corresponds to the greater number and the second one to the less number. A parallel-pipelined version of sorting can be used. Fig.3 shows the diagram of the parallel-pipelined version of exchange sorting of *2N+1* numbers. Using this approach, real time can be reached for the sorting operation. If pixels from local area W with the center in a point *(i,j)* are processed in the step k of the parallel-pipelined exchange sorting algorithm, then the pixels from area W with the center in point *(i,j-1)* are performed simultaneously on the step *k-1* of algorithm, the pixels from area W with the center in point *(i,j-2)* are processed in the step *k-2* and etc.

In the next stage, the averaging of a subset of the ordered pixel sequence

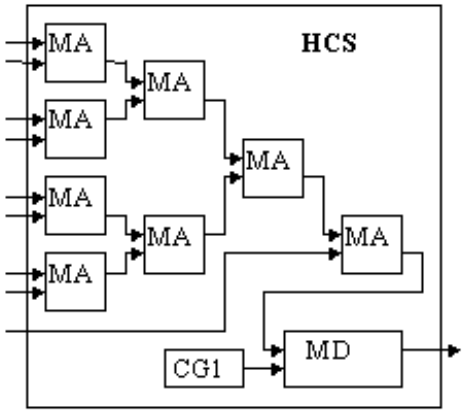

Figure 4: Averaging of sorted sequence for implementation of trimmed mean filter

is calculated, which depends from the trimmed mean filter type: inner, outer, left, right. Fig.4 shows the microprogram modules of addition which fulfils addition in parallel-pipelined mode for a pixel sequence in real time and the microprogram module of division, where divider is equal to the number of pixels of the sequence taken. As far as a small number of cells is needed for the construction of the microprogram module of addition, it is possibility to realize simultaneously all four types of trimmed mean filtering and to switch for one or another filter according to the type of the conditional distribution of noise.

5 Implementation of structure-adaptive filtering and segmentation

The classical structure-adaptive algorithm operates on some image area which is called a window. The window is divided onto smaller subareas which are partly overlapped one with another and contains the central element of the window to be changed on new value, calculated according to its neighbourhood. By a measure of homogeneity of the subarea may serve some local statistical properties, such as mean square deviation or mean absolute deviation. As far as these functions should be calculated for every pixel and the subarea with minimal value should be found, so it is clear, that these procedures require considerable computational cost.

Fig.5 shows the scheme of the implementation of structure-adaptive filtering or segmentation on the homogeneous computing structure. There are

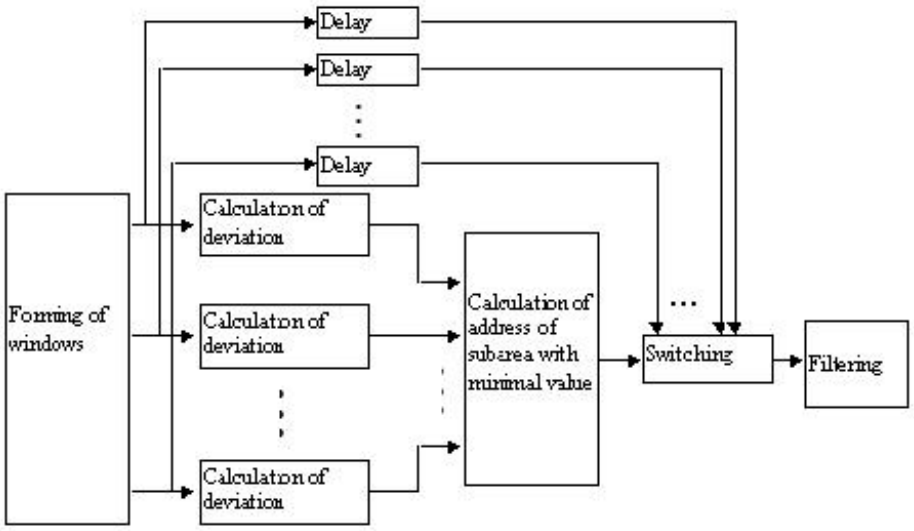

Figure 5: Implementation of structure-adaptive filtering on the homogeneous computing structure

following basic units: the former of a running window and subareas (FRW); calculator of mean square or absolute deviation (CMSD); address calculation of the minimal value of deviation (ACMD); delay lines (DL); the switcher; the filtering unit (FU).

For implementation of structure-adaptive algorithms, the size of the window is ordinary chosen 5×5 or more. The homogeneous computing and storage structure allows to form such running windows and organize the parallel access to all elements of the window to be able simultaneously calculate their statistical properties.

The number of units CMSD is equal to the number of subareas within the running window to meet real time requirements. All these units work in parallel and their results should be brought to the next unit ACMD simultaneously.

Algorithm of exchange sorting is used for finding the minimal value of mean square or mean absolute deviation from n structural subareas. Since, after all, it does not need a value of minimal square deviation, but only the address of image subarea in which this value is detected, so the microprogram module of min/max is exposed to modification and a path for a string with an address of a subarea (from 1 to n) is inserted to microprogram module. This string is toughly tied to the value of mean square or mean absolute deviation and circulates with it according to the scheme of sorting.

The scheme of exchanged sorting can be exposed the shortage, as long as

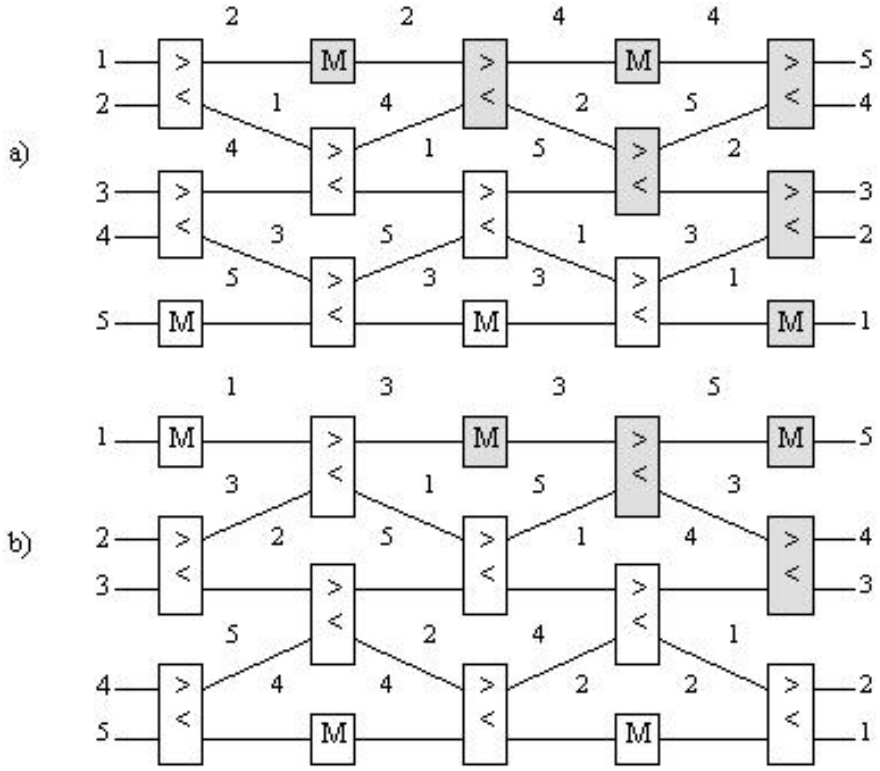

Figure 6: Scheme of exchange sorting with a different order of turn of min/max microprogram modules and memory modules

it computes only the minimal value of deviation. As it can be seen in fig.6, where two schemes of exchanged sorting are shown, they differ by the order of turn for min/max microprogram modules and memory modules. In Fig.6a the scheme of exchanged sorting is shown which is begun in the top left hand corner from the min/max microprogram module and ended in the lower left hand corner with the memory module. This scheme enables to economize more computing cells for finding minimal value (approximately 40%) than scheme shown in Fig.6b (stroked modules may be moved). But, for detection of the maximal value, the scheme shown in Fig.6b is better.

Delay lines are used for compensation of delay caused by units CMSD and ACMD. Pixels of the subarea with the minimal value of deviation are passed to the filtering unit to be filtered by the trimmed mean filter which has been described in previous section.

6 Conclusion

In the paper, the reconfigurable computing architecture based on the homogeneous computing structure concept with possibility of hardware modification in response to changes in processing environment and tuning on implementation of filtering and segmentation algorithms in real time is presented. Configurable computing systems based on the homogeneous computing structure have some essential advantages over configurable computing systems based on field programmable gate arrays, in particular for that algorithms, which can be represented in parallel-pipeline mode.

Using the configurable architecture, it will be able to carry out computations cheaper, faster and consuming less power, so this device can be embedded in a personal computer or workstation and applied for image processing and video compression operations in real time using rapid hardware reconfiguration.

Acknowledgement

This work was supported in part by the project agreement 1931 from the Science and Technology Center in Ukraine.

References

[1] J. Villasenor, B. Schoner, K.-N. Chia, C. Zapata, H. J. Kim, C. Jones, S. Lansing and B. Mangione-Smith, Configurable computing solutions

for automatic target recognition, *4th IEEE Symposium on FPGA's for Custom Computing Machines* (1996), 70-79.

[2] A. DeHon, The density advantage of configurable computing, *Computer 4*, (2000), 41-49.

[3] M. Ya. Bartish, M. P. Bogachev et al., *Parallel Data Processing. Volume 3:Computer Systems, Structures and Media for Solving High-dimensional Problems*, Naukova Dumka, Kyiv, (1986), P. 288.

[4] A. Yu. Lutsyk, Medical image processing and compression on homogeneous computing structure, *54th ICB Seminar, Multimedia, Data Integration, Medical Databases*, Warsaw, (1999), 31-32.

[5] A. Yu. Lutsyk, B. V. Kisil' and O. L. Pelenskyy, Image processing in real time on configurable computing architecture, *The Third International Conference on Digital Information Processing and Control in Extreme Situations*, Minsk, (2002), 124-129.

[6] R. M. Palenichka, P. Zinterhof, Yu. B. Rytsar and I. B. Ivasenko, Structure-adaptive image filtering using order statistics, *Journal of Electronic Imaging 7*, (1988), 339-349.