The object of research is the process of designing hardware devices for sorting arrays of binary data using the methodology of space-time graphs.

The main task that is solved in the work is the development and research of multi-cycle operating devices for sorting binary data in order to choose the optimal structure with predetermined technical characteristics for solving the sorting problem. As an example, the development of different types of structures of multi-cycle operating sorting devices by the method of «even-odd» permutation is shown and their system characteristics are determined.

New structures of multi-cycle operating devices have been designed for a given sorting algorithm, and analytical expressions for calculating equipment costs and their performance have been given. A comparative analysis of the hardware and time complexity of the developed structures of devices for sorting binary numbers of various types with known implementations of algorithmic and pipeline operating devices was carried out. As a result, the proposed structures, when sorting large arrays of binary data (N>128), have an order of magnitude less hardware complexity due to sequential execution of the same type of operations. The time complexity of multi-cycle operating devices of combined and sequential types with large values of input data is 2.3 and 3.4 times less than that of known pipeline operating devices.

A feature of the research results is the possibility of finding the optimal ratio between the hardware and time characteristics of the resulting structures of sorting devices. Owing to this, the designer will be able to choose the necessary type of device for the implementation of the corresponding task with optimal system characteristics.

The field of application of the designed sorting devices is the tasks of digital processing of signals and images. The practical use of the developed sorting devices can be carried out in the form of their synthesis on software integrated logic circuits

Keywords: «even-odd» permutation method, multi-cycle operating device, space-time graph, flow graph, algorithm, synthesis, functional operator, sorting device, binary data, simulation

UDC 621.518
DOI: 10.15587/1729-4061.2023.285997

DESIGN OF VARIOUS OPERATING DEVICES FOR SORTING BINARY DATA

Volodymyr Hryha
Corresponding author
PhD, Associate Professor*
E-mail: volodymyr.gryga@pnu.edu.ua
Bogdan Dzundza
PhD, Associate Professor*
Stepan Melnychuk
Doctor of Technical Sciences, Professor**
Iryna Manuliak
PhD, Associate Professor**
Andriy Terletsky
PhD, Associate Professor*
Mykhailo Deichakivskyi
Postgraduate Student*
*Department of Computer Engineering and Electronics
Vasyl Stefanyk Precarpathian National University
Shevchenko str., 57, Ivano-Frankivsk, Ukraine, 76018
**Department of Computer Systems and Networks
Ivano-Frankivsk National Technical University of Oil and Gas
Karpatska str., 15, Ivano-Frankivsk, Ukraine, 76019

Received date 14.06.2023
Accepted date 18.08.2023
Published date 31.08.2023

How to Cite: *Hryha, V., Dzundza, B., Melnychuk, S., Manuliak, I., Terletsky, A., Deichakivskyi, M. (2023). Design of various operating devices for sorting binary data. Eastern-European Journal of Enterprise Technologies, 4 (4 (124)), 6–18. doi: https://doi.org/10.15587/1729-4061.2023.285997*

1. Introduction

Sorting is one of the typical tasks of data processing [1, 2] and is usually understood as the task of arranging elements of an unordered set of values of data arrays in monotonically increasing or decreasing order [3, 4]. The sorting operation takes an average of 25 % of machine time [1] and is most often used in digital signal and image processing tasks and distributed wireless sensor networks [5]. There are software and hardware techniques of implementing the sorting operation [6, 7]. Software methods of implementing sorting algorithms are currently fairly well described and researched in literary sources [1, 4, 6]. As for the hardware techniques of implementing sorting algorithms, the structures of sorting devices with the use of spatial and temporal parallelism are currently quite well described in [8, 9].

Spatial parallelism is understood as parallel execution of sorting algorithm operations by several operating devices (ODs) at the same time, which significantly speeds up the execution time of the algorithm [8, 10, 11]. It should be noted that parallel execution of operations is used for algorithms in which the sequence of performed operations depends only on the number of input data.

Time parallelism is understood as combining the operations of the sorting algorithm in time by dividing operational blocks by conveyor registers, which significantly speeds up the performance of the operating device [8, 12].

However, the use of algorithmic and conveyor sorting devices for multi-bit arrays of binary data has important drawbacks – high hardware and structural complexity of OD. Also, in the case of one-time use of such devices for the implementation of the sorting algorithm, a significant part of

the equipment may have a low percentage of use and remain unused for a long time.

Therefore, the simultaneous combination of spatial and temporal parallelism for the purpose of choosing the optimal structure relative to the system characteristics is an urgent issue for the construction of sorting ODs.

Thus, an important scientific and applied task is the development of various types of binary data sorting devices and the study of their system characteristics using space-time parallelism. It should be noted that with non-critical time parameters, the hardware complexity of the device can be significantly reduced, as a result of which the area of FPGA crystal will decrease.

The task set is solved by applying the methodology of space-time graphs (STG) [13, 14], which allows obtaining different types of structures of multi-cycle operating sorting devices [15, 16]. The evaluation of the system characteristics of the resulting hardware structures will allow the designer to choose the optimal structure relative to the given technical parameters to solve the necessary problem.

Therefore, research aimed at developing hardware devices for sorting arrays of binary data based on STG is relevant in the field of designing computer systems and their components with improved system characteristics.

2. Literature review and problem statement

In [9], a comparison of parallel sorting networks based on parallel sorting algorithms was considered, and their comparative analysis was carried out. The dependences of the efficiency of sorting network structures have been determined, based on the results of which it is possible to choose one or another sorting network for implementation. At the same time, the traditional methodology for building such sorting networks based on flow graphs of the algorithm was used, which have limitations regarding the choice of the optimal structure.

In [10], various options for the synthesis of parallel computing sorting devices are considered depending on the amount of input data and the width of the parallel form of the algorithm. With a fixed number of input values, with a decrease in the width of the algorithm flow graph (AFG), its height increases within the range from the minimum height value (number of tiers) to the maximum value (number of functional operators) of AFG.

By changing the degree of parallelization of algorithms, the complexity of their hardware implementation does not change but only the algorithm execution time changes. The disadvantage of this approach is the functional limitations of using such algorithms. The given results do not provide an opportunity to evaluate the data structure of the sorting devices when the input data is fully loaded.

In [11], an analysis of methods for parallel sorting of arrays of binary data was carried out. As a result of the research, it was determined that the algorithm for implementing the method of parallel sorting of data by merging, compared to the algorithms for sorting numbers by counting, displacement, and insertion, is more structured, more uniform, and more oriented towards parallel-conveyor implementation. Merge sort algorithms are based on the basic operation of combining two or more ordered arrays into one ordered array. The hardware implementation of the basic operation of combining three or more ordered arrays into one ordered array is complex and requires significant hardware costs. Simpler is the basic operation of merging two ordered arrays into one ordered array, that is, a two-way merge. The disadvantage of existing algorithms for implementing two-way merging is low performance since they are all based on operations of pairwise comparison of data elements.

In [12], highly efficient parallel structures were developed for sorting intensive streams of binary data in real time using an improved merge method. Due to the change in the number of channels of the OD structure and the bit rate of the input data, it was possible to achieve coordination of the intensity of data arrival with the sorting ability, which increased the indicator of the efficiency of equipment use. However, the devised sorting method is focused on the architecture of graphics processors, which requires additional resource-intensive costs.

In [17], a new adaptive OneByOne sorting algorithm is proposed, which, compared to choice sorting and bubble sorting algorithms, has a faster operating time for different array sizes. The research results regarding the proposed algorithm show its effectiveness in software implementation on a modern element base. However, the issue of hardware implementation of such algorithms remains unresolved in the paper. The study of the system characteristics of operating devices during the hardware implementation of adaptive sorting algorithms would provide an opportunity to reveal the depth of this issue.

In [18], a new algorithm for sorting n k-bit binary numbers is proposed and its hardware is described. The hardware implementation of this algorithm is based on the use of shift registers, counters, and logic elements. The disadvantage of this algorithm is a significant number of iterations (cycles) – $(n+2k)$ clock cycles, which ultimately requires significant hardware costs when implemented on FPGA. The issue of reducing the number of cycles by changing the structure of the algorithm is not given enough attention in the cited paper.

A new library for sorting binary data is described in [19]. The advantage of such a library is the ability to sort floating point numbers. The hardware implementation of such a library improves the throughput of the system. However, the clock frequency does not have a significant increase, and the hardware resources are twice as large as compared to the known ones. However, the possibility of reducing the hardware complexity of comparison and exchange blocks is not considered in the work.

Paper [20] describes hardware sorting using parallel recursive algorithms based on binary trees. The hardware implementation is carried out using memory blocks and finite state automata, which ultimately requires significant hardware costs when implemented on FPGA. The issue of reducing hardware costs using a simpler element base is not considered in the cited work.

In the analysis of available studies, the issue of researching multi-cycle ODs for data sorting was not given enough attention.

All this gives reason to assert that it is expedient to conduct a study of various structures of sorting devices, which are focused on adaptive parallel algorithms using the methodology of space-time graphs [13, 14]. This methodology takes into account the specificity of building multi-cycle operating devices taking into account spatial and temporal characteristics. That is why the implementation of highly efficient multi-cycle devices for sorting arrays of binary numbers requires extensive use of the modern element base, improvement of existing and development of new device structures.

Therefore, the problem of building effective multi-cycle ODs for sorting arrays of binary data, taking into account the spatial and temporal characteristics of their operation,

remains unsolved. Solving this task could give the designer the opportunity to choose the structure of multicycle OD sorting that is optimal in terms of hardware costs and efficient in terms of time characteristics.

3. The aim and objectives of the study

The purpose of this work is to design multi-cycle operating devices for sorting binary data, which, given non-critical time parameters, make it possible to reduce the hardware complexity of the device and the area of the crystal that the developed system will occupy. This would make it possible to significantly reduce the costs of hardware implementation of such devices and, in some cases, to improve time characteristics due to a combined approach.

To achieve the goal, the following tasks were solved:

– to perform a hardware implementation of the sorting algorithm by the method of «even-odd» permutation by building an algorithmic and pipeline structure based on the algorithm flow graph and investigate their system characteristics using an improved comparison scheme;

– to perform a hardware implementation of the sorting algorithm by the method of «even-odd» permutation by building various types of multi-cycle operating devices (MOD) using space-time graphs and investigate their system characteristics;

– to perform a practical implementation on FPGA for various types of binary data sorting devices.

4. The study materials and methods

The object of our study is the process of designing multi-cycle operational sorting devices using the methodology of space-time graphs [13–15].

The hypothesis of the study assumed the possibility of using the theory of algorithm flow graphs [8], with the help of which the designer performs a direct mapping of the received graph into the structure of the device, in order to develop multi-cycle operating devices. In [9], it is proposed to use a multi-channel sorting memory to combine functional operators of the same type. This memory has a number of advantages in terms of speed when working with digital signal and image processing algorithms. However, the assumptions were adopted that for the implementation of adaptive algorithms of mathematical operations, the use of sorting memory significantly increases the hardware costs of the device. It is also necessary to build a device for managing such memory, which in turn further increases the hardware complexity of the resulting device.

In order to improve such shortcomings, it was decided to use the methodology of space-time graphs in the construction of multi-clock operating sorting devices. This methodology makes it possible to implement different types of operating devices with an optimal ratio of hardware and time complexity by combining operations of the same type. Simple components are used to build multi-cycle ODs on the basis of STG: multiplexers, demultiplexers, operational blocks and delay elements of intermediate results (registers). The use of simple components makes it possible to significantly reduce hardware costs for the construction of the resulting device.

The theory of complexity [8] was applied to study the system characteristics of operational sorting devices, in particular, the hardware and time complexity of various types of ODs were analyzed. Analytical expressions for calculating the hardware and time complexity of the presented structures of single-cycle and multi-cycle ODs were derived. The description of the models of the implemented structures of the sorting OD at the behavioral level was performed using the hardware description language – VHDL. Modeling of the developed sorting OD structures was performed in the integrated Active-HDL environment of Aldec (USA). The synthesis of single-cycle and multi-cycle ODs for binary data sorting was carried out on the Xilinx (USA) FPGA [21–23], Artix7 family, in the Vivado Design Suite environment [21–23]. As a result of the research, the obtained theoretical calculations were compared with the results of practical implementation of various types of binary data sorting ODs.

5. Results of investigating hardware devices for sorting arrays of binary data

5. 1. Studying the algorithmic and conveyor operational sorting devices

Fig. 1 shows the graph of the algorithm for sorting «even-odd» permutation for 6 input values [9, 10, 16].

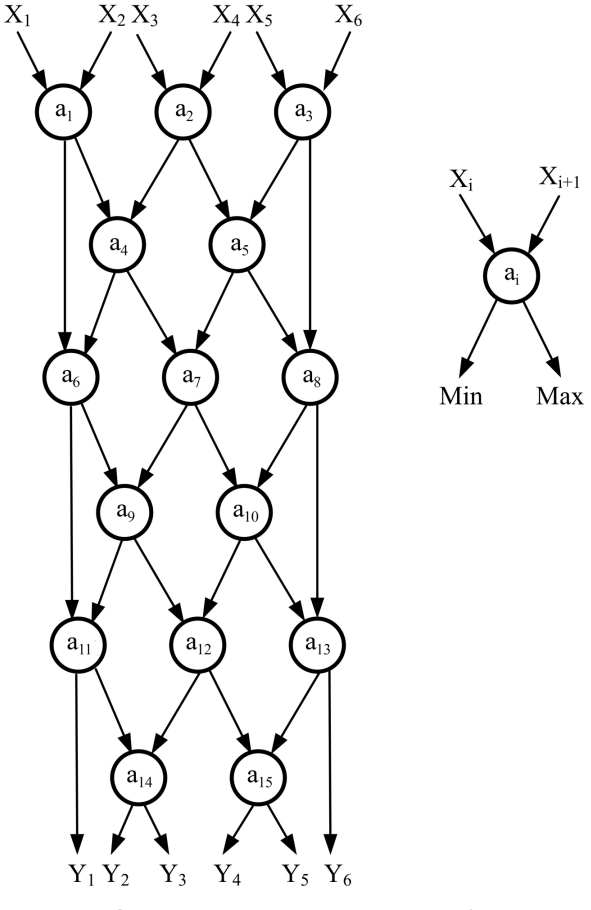

Fig. 1. Graph of the algorithm for sorting 6 values by the method of «even-odd» permutation

The sorting algorithm using the «even-odd» permutation method requires $N(N-1)/2$ «compare and rearrange» operations for N input values. The time complexity of sorting by the «even-odd» method is N operations «compare and rearrange» [1, 8–10, 16]. The width of FG of the sorting algorithm by the method of «even-odd» permutation is $N/2$, and the height is N of «compare and rearrange» operations.

To build the structure of the algorithmic device, it is necessary to perform a direct mapping of the graph elements of the flow graph into the structure of the operating device. That is, the vertices of the graph (functional operators) will correspond to a hardware component or module, and to the arcs – lines for transmitting input data, intermediate and final results.

Fig. 2 shows the structure of the algorithmic device for sorting 6- and 8-digit numbers by the method of «even-odd» permutation.

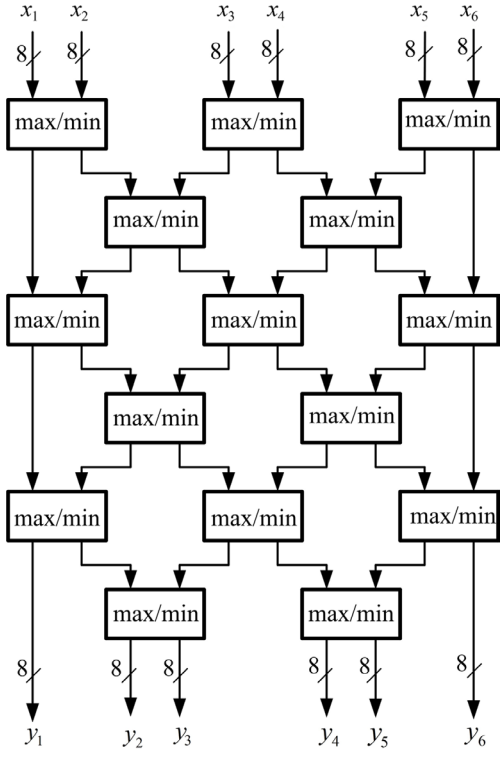

Fig. 2. The structure of the algorithmic operating sorting device by the method of «even-odd» permutation for 6 input values

Fig. 3 shows a block diagram of the basic «compare and rearrange» operation used in binary data sorters. This operation allows one to compare two numbers «by more» and output the maximum and minimum value of the compared numbers to the corresponding outputs [9, 13, 16].

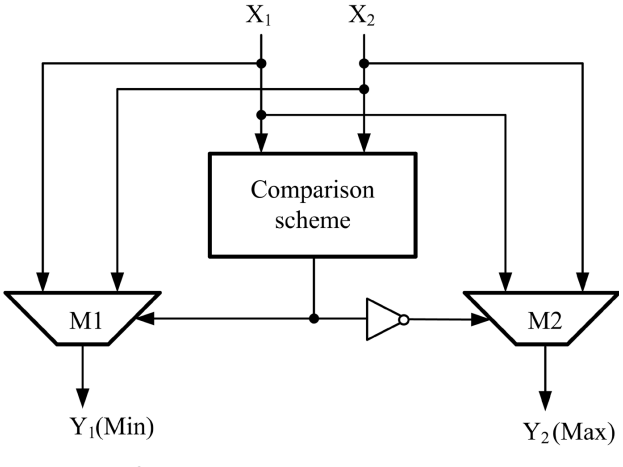

Fig. 3. Block diagram of the basic operation of the «even-odd» permutation

The structural diagram of the «compare and rearrange» operation consists of a comparator (comparison circuit) and two multiplexers (*M*1, *M*2). The comparator compares two numbers according to the corresponding sign («by more»). If the number ($X_1>X_2$), «1» is formed, and if the number ($X_1<X_2$) – «0». Depending on the received value of the output signal from the output of the comparator («0» or «1»), the multiplexer (M1) outputs a smaller number to the output (Y_1), and the multiplexer (*M*2) outputs a larger number to the output (Y_2).

The hardware complexity of a 2-input single-bit multiplexer is: $A_{Mux}=5$ (gates), and the time complexity is equal to: $t_{Mux}=3$ (microclocks).

In [27], an improved structure of the «more than» comparison scheme using one-bit non-positive binary adders is proposed.

The structure of the improved comparison scheme for two 4-bit numbers is shown in Fig. 4.

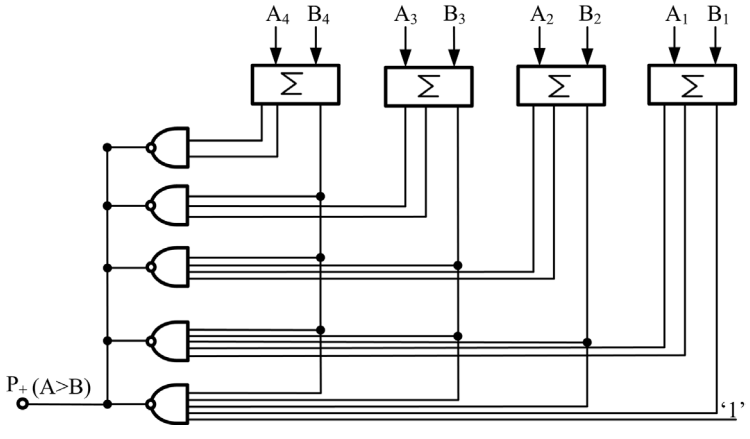

Fig. 4. The structure of the scheme for comparing two 4-bit numbers «for more»

The scheme for comparing multi-bit binary data works as follows: direct multi-bit data codes from the input 2*n*-bit bus are fed to the inputs of single-bit incomplete binary adders. Signals from the first and second outputs of one-bit incomplete binary adders are fed to the corresponding inputs of NAND logic elements. At the same time, if a logical «1» signal is formed at all inputs of one of the logical «NOT-AND» elements, a logical «0» signal is formed at the output channel of the device ($\overline{P_+}$), which corresponds to the comparison condition ($A>B$). Otherwise, a logical «1» signal is generated at the output of the device, which corresponds to the comparison condition ($A<B$).

The hardware complexity of the above comparison scheme is $A_{CS}=13$ (gates), and the time complexity is equal to $t_{CS}=2$ (microcycles).

In comparison with the known comparison scheme [9, 13, 16], the proposed scheme has 1.8 times lower hardware complexity and 3 times faster performance.

The hardware complexity of the algorithmic operating device for sorting by the method of «even-odd» permutation will be calculated according to the following formula:

$$A_{AOD} = N(N-1)/2\times(A_{CS}+2A_{Mux}), \quad (1)$$

where A_{CS} is the hardware complexity of the *n*-bit comparison circuit, A_{Mux} is the hardware complexity of the 2-input *n*-bit multiplexer, and *N* is the number of input data.

For example, with N=8 and n=4, the hardware complexity of the algorithmic OD will be equal to A_{AOD}= =28×(13+2×28)=1932 (gates).

The time complexity of the algorithmic sorting device will be equal to:

$$T_{AOD} = N \times (t_{CS} + t_{Mux}), \quad (2)$$

where t_{CS} is the time complexity of the n-bit comparison circuit, t_{Mux} is the time complexity of the 2-input n-bit multiplexer, N is the number of input data.

For example, with N=8 and n=4, the time complexity of the algorithmic OD will be equal to T_{AOD}=8×(2+3)=40 (microcycles).

The throughput of the sorting algorithmic operating device will be equal to the time complexity of the device $Th_{AOD}=T_{AOD}$.

Disadvantages of algorithmic ODs include high hardware complexity and low utilization of equipment [8, 13, 16]. This is especially true in the case of one-time data transfer; other elements of such devices remain idle and use significant system resources.

The pipeline principle of processing involves the combination of operators of sorting algorithms on different data in the execution time [8, 16].

Fig. 5 shows the pipeline structure of the device for sorting 6- and 8-bit numbers by the method of «even-odd» permutation, which contains pipeline registers. Input registers are designed to record input data, registers placed between hardware modules store intermediate calculation results, and output registers record the final result.

The hardware complexity of the conveyor operating device sorting by the method of «even-odd» permutation will be calculated according to the following formula:

$$A_{COD} = A_{AOD} + N(N+1)A_{R_{conv}}, \quad (3)$$

where A_{AOD} is the hardware complexity of the algorithmic operating device, $A_{R_{conv}}$ is the hardware complexity of the n-bit pipeline register, and N is the number of input data.

For example, with N=8 and n=4, the hardware complexity of the conveyor OD will be equal to A_{COD}=1932+(72×16)= =3084 (gates).

The time complexity of this sorting device will be equal to:

$$T_{COD} = T_{AOD} + (N-1) \times t_{R_{conv}}, \quad (4)$$

where T_{AOD} is the time complexity of the algorithmic operating device, $t_{R_{conv}}$ is the time complexity of the n-bit pipeline register, and N is the number of input data.

For example, with N=8 and n=4, the time complexity of the pipeline OD will be equal to T_{COD}=40+(7×2)=54 (microcycles).

The throughput of the sorting conveyor operating device will be equal to:

$$Th_{COD} = t_{CS} + t_{Mux} + t_{R_{conv}}. \quad (5)$$

Disadvantages of pipeline ODs include high hardware complexity due to the use of pipeline registers. The advantage of pipeline ODs is high data processing performance [9, 16] and low bandwidth, which is determined using formula (5).

Fig. 6 shows the plot of dependence of the number of logical elements (gates) on the amount of input data for the algorithmic and pipeline OD sorting algorithm by the method of «even-odd» permutation, built on the basis of analytical expressions (1) and (3).

Fig. 7 shows a plot of dependence of the total number of microcycles on the amount of input data for algorithmic and conveyor operational devices of the sorting algorithm by the method of «even-odd» permutation built on the basis of analytical expressions (2) and (4).

As can be seen from the graphic dependences, the pipeline OD has 1.2 times greater hardware complexity than the algorithmic operating device due to the pipeline registers. The time complexity of the conveyor OD is 1.5 times less than that of the algorithmic OD, which allows one to significantly speed up the execution of the operations of the sorting algorithm when the conveyor is fully loaded.

Algorithmic and pipeline ODs have some significant limitations, which are that the designer can build only one type of such devices, which will have stable system characteristics.

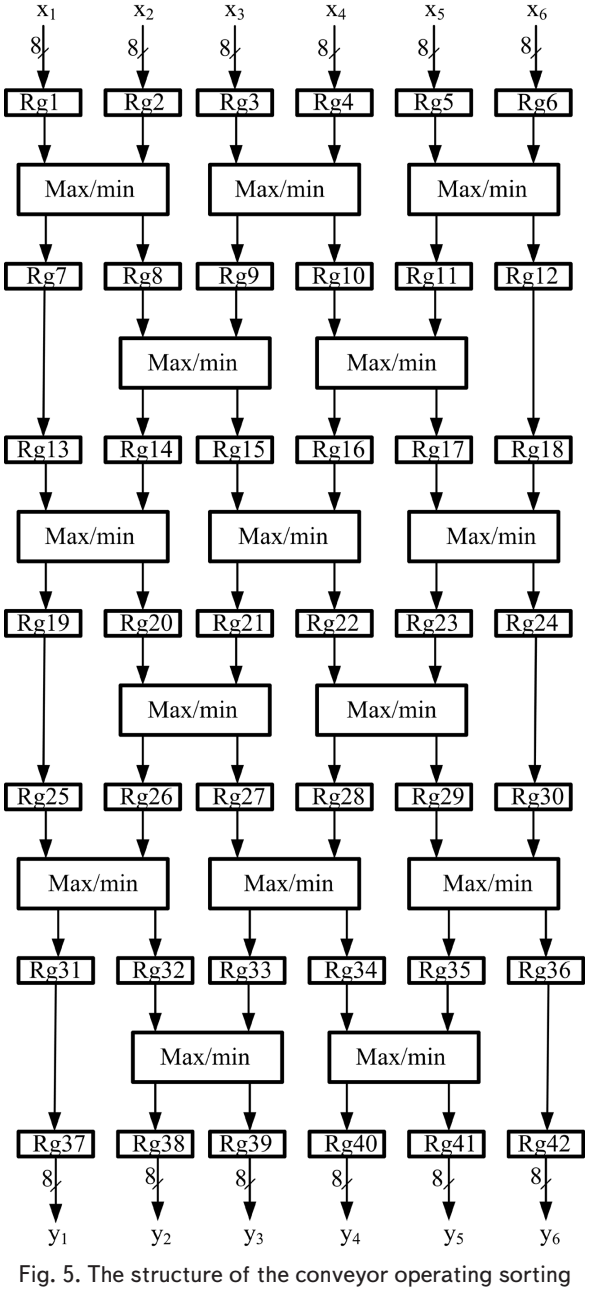

Fig. 5. The structure of the conveyor operating sorting device by the method of «even-odd» permutation for 6 input values

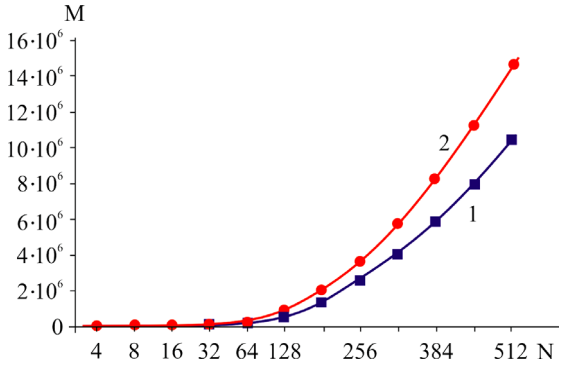

Fig. 6. Dependence plot of the total number of gates *M* on the amount of input data *N* for algorithmic (curve 1) and pipeline (curve 2) operating devices

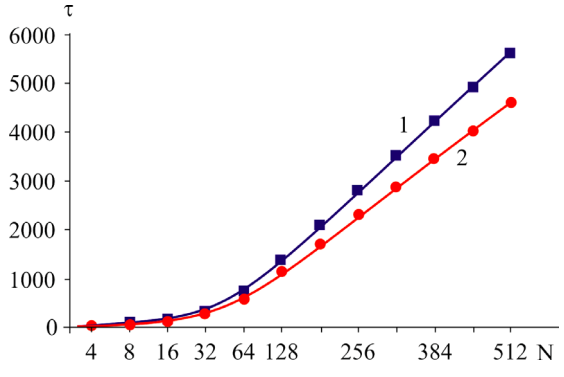

Fig. 7. Dependence plot of the total number of microcycles τ on the amount of input data *N* for algorithmic (curve 1) and pipeline (curve 2) operating devices

That is why the construction of various types of multi-cycle ODs for data sorting gives the designer the opportunity to analyze the dynamics of changes in system characteristics. Namely, it becomes possible to find the optimal ratio between hardware and time complexity, which allows one or another type of device to be used to solve a problem with predetermined conditions.

5. 2. Research of multi-cycle operating sorting devices

To implement various types of multi-cycle operating devices (MODs) for sorting by the method of «even-odd» permutation, it is necessary to transform AFG into the corresponding space-time graphs that correspond to various types of such MODs [8, 14–16].

Fig. 8 shows the structure of MOD of the combined type of the sorting algorithm by the method of «even-odd» permutation.

This MOD structure consists of input registers, 5 operational units that perform the same basic operations of the algorithm sequentially in time, multiplexers, demultiplexers, delay registers for intermediate results, and output registers. In the registers (Rg7, ..., Rg12), intermediate results are delayed by the corresponding number of cycles. The multiplexers (M1–M6) sequentially pass the input data and intermediate results to the inputs of the operational units using the values of the control signals (sm1, ..., sm6) generated by the control device. Demultiplexers (Dm1–Dm6) distribute the intermediate results to the inputs of the operational units at the required time, and the final results are recorded in the output registers (Rg13, ..., Rg18).

Equipment costs for the implementation of a combined type MOD for a given sorting algorithm will be equal to:

$$A_{CMCOD} = (N-1)A_{BS} + NA_{Rin} + NA_{Mux(2\to1)} + \\ + NA_{Dmux(1\to2)} + NA_{IRDE} + NA_{Rout}, \quad (6)$$

where A_{BS} is the hardware complexity of the n-bit binary number comparison scheme; A_{Rin} is the hardware complexity of input n-bit registers; A_{Rout} – hardware complexity of output n-bit registers; A_{Mux} – hardware complexity of multiplexers; A_{Dmux} – hardware complexity of demultiplexers; A_{IRDE} is the hardware complexity of the intermediate result delay registers.

According to formula (6), the combined MOD consists of (N–1) computing units, N input and output n-bit registers. Also, the MOD structure contains N intermediate n-bit registers, N multiplexers with 2 inputs, and N demultiplexers with 2 outputs.

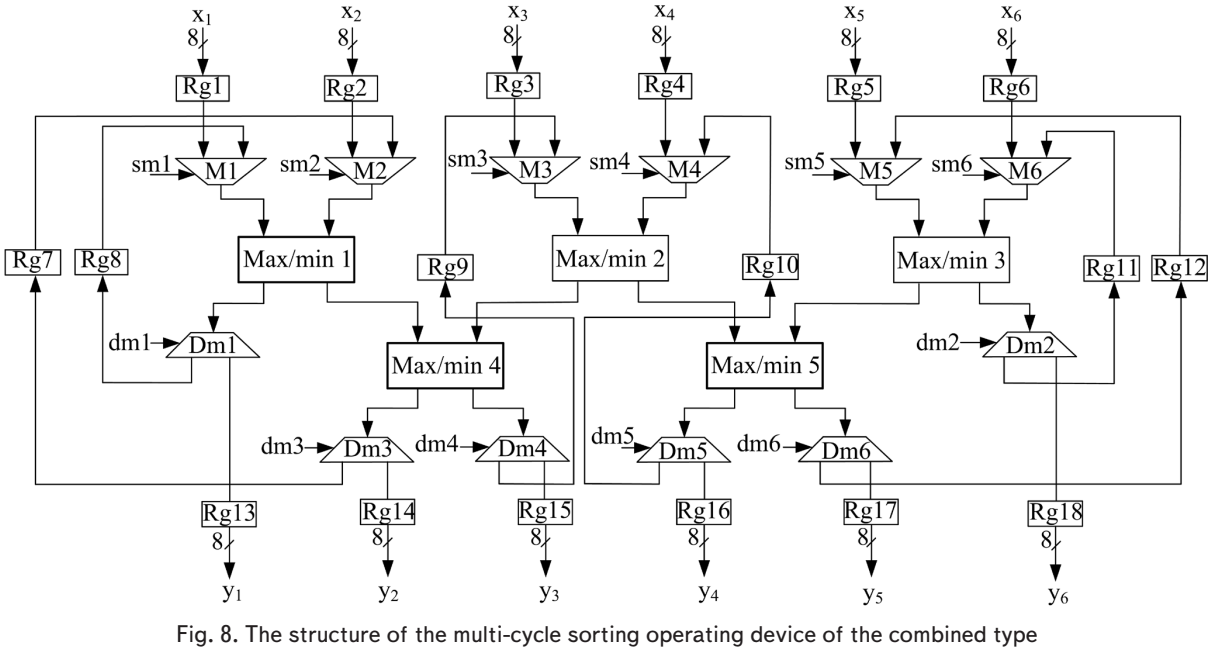

Fig. 8. The structure of the multi-cycle sorting operating device of the combined type

The time complexity of the combined MOD sorting will be equal to:

$$T_{CMCOD} = k_1 \times \left(t_{BS} + t_{Mux} + t_{Dmux} + t_{Reg}\right), \quad (7)$$

where t_{BS} is the time complexity of the n-bit comparison scheme, t_{Mux} is the time complexity of the multiplexer, t_{Dmux} is the time complexity of the demultiplexer, t_{Reg} is the time complexity of the delay registers of intermediate results, k_1 is the number of iterations, which is equal to:

$$k_1 = \frac{N(N-1)/2}{N/2+(N/2-1)}.$$

For example, with N=8 and n=4, the time complexity of the combined MOD will be equal to T_{CMCOD}=4×(9+3+2+2)= =64 (microcycles).

The throughput of the combined MOD sorting will be equal to the time complexity of the device $Th_{CMCOD}=T_{CMCOD}$.

The methodology for obtaining this MOD structure using iterative STG is shown in [16].

Fig. 9 shows the structure of MOD of the iterative type of the sorting algorithm by the method of «even-odd» permutation.

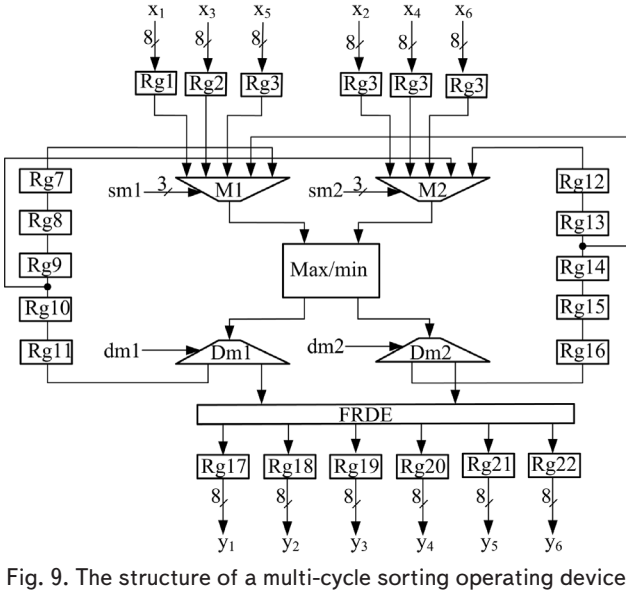

Fig. 9. The structure of a multi-cycle sorting operating device of the iterative type

This structure of MOD consists of input registers, one operational unit that performs all the same operations of the algorithm sequentially in time, multiplexers, demultiplexers, delay registers for intermediate and final results, and output registers.

The hardware complexity of MOD of the iterative type for the sorting algorithm by the «even-odd» permutation method will be equal to:

$$A_{IMCOD} = A_{BS} + 2\left(A_{Mux\left(\left(\frac{N}{2}+2\right)\to 1\right)} + A_{Dmux(1\to 2)}\right) + \\ + A_{IRDE} + \left(\frac{N}{2}-1\right) + (N-2)A_{FRDE} + 2NA_{Rin/Rout}, \quad (8)$$

where A_{BS} is the hardware complexity of the n-bit binary number comparison scheme; $A_{Rout/Rin}$ – hardware complexity of n-bit input/output registers; A_{Mux} – hardware complexity of multiplexers; A_{Dmux} – hardware complexity of demultiplexers; A_{IRDE}=(2N–2) – hardware complexity of n-bit delay registers for intermediate results; A_{FRDE} is the hardware complexity of n-bit final result delay registers.

According to formula (8), a MOD of the iterative type consists of one computing unit, N input and output n-bit registers. Also, the MOD structure contains (2N–2) intermediate result delay registers, two (N/2+2)-input multiplexers, two demultiplexers, and (N/2–1)+(N–2) final result delay registers.

The time complexity of the iterative sorting MOD will be equal to:

$$T_{IMCOD} = k_2 \times \left(t_{BS} + t_{Mux} + t_{Dmux} + t_{Reg}\right), \quad (9)$$

where t_{BS} is the time complexity of the n-bit binary number comparison scheme; t_{Mux} – time complexity of multiplexers; t_{Dmux} – time complexity of demultiplexers; t_{Reg} is the time complexity of registers delaying intermediate results; k_2 is the number of iterations, which is equal to the number of algorithm operations: $k_2=N(N-1)/2$.

For example, with N=8 and n=4, the time complexity of the iterative MOD will be equal to T_{IMCOD}=28×(9+3+2+2)= =448 (microcycles).

The throughput of the iterative sorting MOD will be equal to the time complexity of the device $Th_{IMCOD}=T_{IMCOD}$.

Fig. 10 shows the structure of the MOD of the sequential-iterative type of the sorting algorithm by the method of «even-odd» permutation.

This structure of MOD consists of input registers, three operational units that perform all the same type of algorithm operations sequentially in time, multiplexers, demultiplexers, delay registers for intermediate and final results, and output registers.

Equipment costs for the implementation of sequential-iterative MOD for a given sorting algorithm will be equal to:

$$A_{SIMCOD} = A_{BS} + (N-2)A_{Mux(3\to 1)} + A_{Mux(2,4\to 1)} + \\ + (N-2)A_{Dm(1\to 3)} + A_{Dm(1\to 2)} + A_{IRDE} + A_{Rin/Rout}, \quad (10)$$

where A_{BS} is the hardware complexity of the n-bit binary number comparison scheme; $A_{Rin/Rout}$ – hardware complexity of n-bit input/output registers; A_{Mux} – hardware complexity of multiplexers; A_{Dmux} – hardware complexity of demultiplexers; A_{IRDE} is the hardware complexity of n-bit intermediate result delay registers.

According to formula (10), a MOD of the iterative type consists of N/2 computing units, N input and output n-bit registers. Also, the MOD structure contains (N–2) 3-input multiplexers, one 2-input and 4-input multiplexer, (N–2) three-output demultiplexers, one two-output demultiplexer and (2N–1) intermediate delay registers results.

The time complexity of sequential-iterative sorting MOD will be equal to:

$$T_{SIMCOD} = k_3 \times \left(t_{BS} + t_{Mux} + t_{Dmux} + 2t_{Reg}\right), \quad (11)$$

where t_{BS} is the time complexity of the n-bit binary number comparison scheme; t_{Mux} – time complexity of multiplexers; t_{Dmux} – time complexity of demultiplexers; t_{Reg} is the time complexity of registers delaying intermediate results; k_3 is the number of iterations, which is equal to:

$$k_3 = \frac{N(N-1)/2}{N/2}.$$

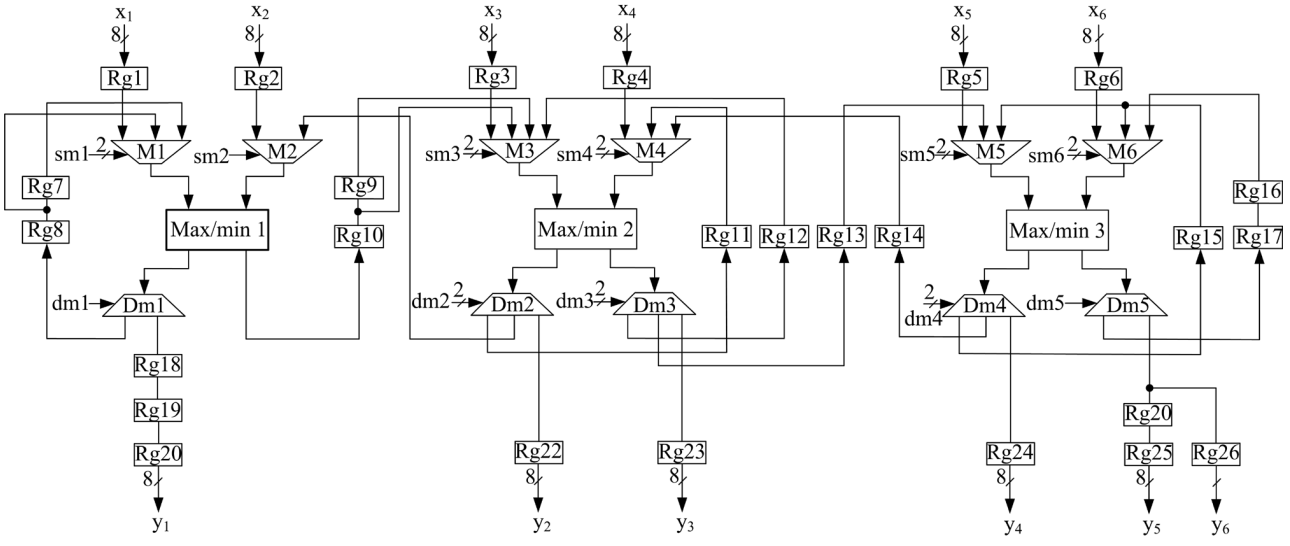

Fig. 10. The structure of a multi-cycle sorting operating device of sequential-iteration type

For example, with N=8 and n=4, the time complexity of the sequentially iterative MOD will be equal to T_{SIMCOD}= =7×(9+3+2+4)=72 (microcycles).

The throughput of sequential-iterative sorting MOD will be equal to the time complexity of the device $Th_{SIMCOD}=T_{SIMCOD}$.

Fig. 11 shows the structure of MOD of the sequential type of the sorting algorithm by the method of «even-odd» permutation.

This MOD structure consists of input registers, 6 operational units that perform all operations of the same type of the algorithm sequentially in time, multiplexers, demultiplexers, delay registers for intermediate and final results, and output registers.

Equipment costs for the implementation of a sequential type MOD for a given sorting algorithm will be equal to:

$$A_{SMCOD} = NA_{BS} + 2NA_{Rin/Rout} + 2NA_{Mux(N/2\to1)} + \\ +(N-2)A_{Mux(2\to1)} + NA_{Dmux(1\to2)} + A_{I/FRDE}, \quad (12)$$

where A_{BS} is the hardware complexity of the n-bit binary number comparison scheme; $A_{Rin/Rout}$ – hardware complexity of n-bit input/output registers; A_{Mux} – hardware complexity of multiplexers; A_{Dmux} – hardware complexity of demultiplexers; A_{IRDE} is the hardware complexity of n-bit delay registers for intermediate results; A_{FRDE} is the hardware complexity of n-bit final result delay registers.

According to formula (12), a MOD of the iterative type consists of N computing units, N input and output n-bit registers. Also, the MOD structure contains two (N/2)-input multiplexers, (N–2) 2-input multiplexers, N demultiplexers, (2N–3) intermediate result delay registers and (N–2) final result delay registers.

The time complexity of sequential sorting MOD will be equal to:

$$T_{SMCOD} = Nt_{BS} + \frac{N}{2}t_{Mux} + \frac{N}{2}t_{Dmux} + Nt_{Reg}, \quad (13)$$

where t_{BS} is the time complexity of the n-bit binary number comparison scheme; t_{Mux} – time complexity of multiplexers; t_{Dmux} – time complexity of demultiplexers; t_{Reg} is the time complexity of registers delaying intermediate results; N is the number of input data.

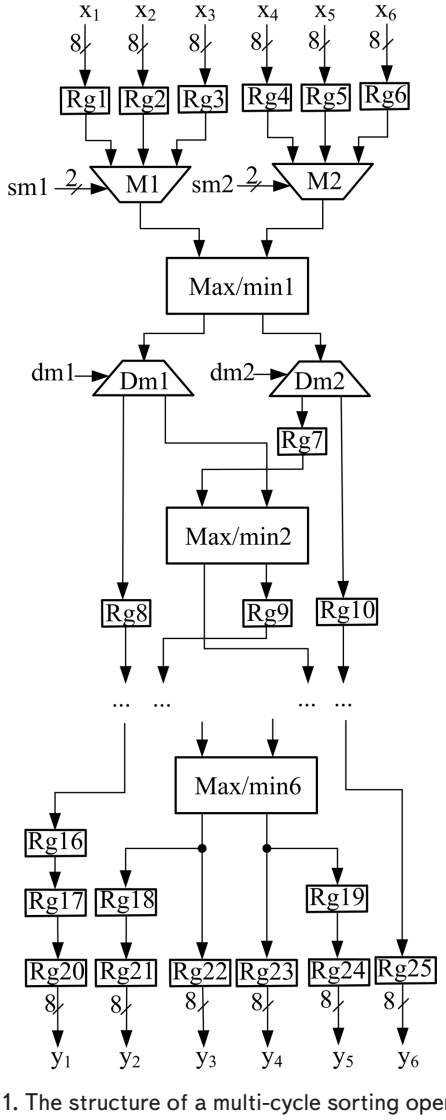

Fig. 11. The structure of a multi-cycle sorting operating device of serial type

For example, with N=8 and n=4, the time complexity of a sequential MOD will be equal to T_{SMCOD}=(8×9+4×3+ +4×2+8×2)=108 (microcycles).

The throughput of sequential sorting MOD will be equal to $Th_{SMCOD}=t_{BS}+t_{Mux}+t_{Dmux}+t_{Reg}$.

Fig. 12 shows a plot of dependence of the total number of gates on the amount of input data for various types of MOD calculated on the basis of the obtained analytical expressions (6), (8), (10), (12).

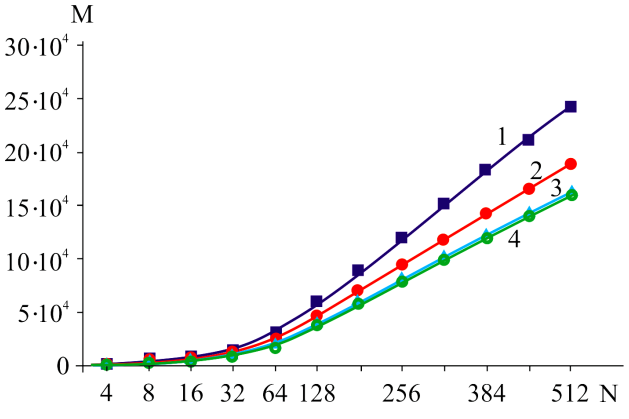

Fig. 12. Dependence plots of the total number of gates M on the amount of input data N for different types of multi-clock operating devices: MOD of the sequential type – curve 1, MOD of the sequential-iterative type – curve 2, MOD of the iterative type – curve 3, MOD of the combined type – curve 4

As can be seen from the graphic dependences, the lowest hardware complexity among different types of sorting MOD is demonstraed by the iterative and combined MODs due to the minimum number of hardware modules and registers.

Fig. 13 shows a plot of dependence of the total number of microcycles on the amount of input data for various types of MODs calculated on the basis of the obtained analytical expressions (7), (9), (11), and (13).

As can be seen from the graphical dependences, the combined MOD has the greatest time complexity among different types of sorting MODs due to the regularity of the structure. Iterative MOD has the lowest speed due to sequential execution of operations.

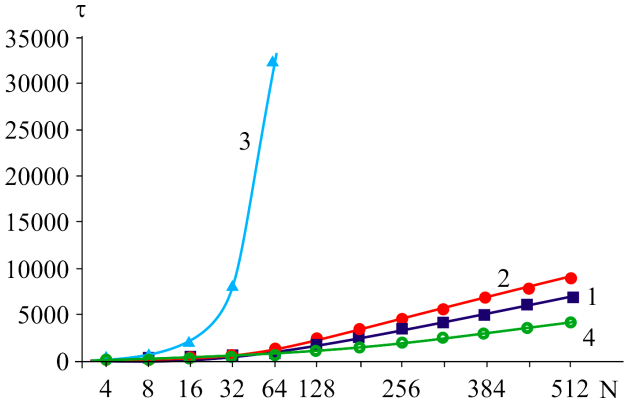

Fig. 13. Dependence plots of the total number of microcycles t on the value of the input data N for different types of multi-cycle operating devices: MOD of the sequential type – curve 1, MOD of the sequential-iterative type – curve 2, MOD of the iterative type – curve 3, MOD of the combined type – curve 4

5. 3. Realization and research of obtained structures of sorting devices on programmable integrated circuits

The designed structures of sorting devices were programmatically described using the VHDL hardware description language and synthesized on the Xilinx FPGA using the Vivado automated design system. Fig. 14 shows the block diagram of the combined-type MOD.

In this block diagram, one can see 6 input 8-bit buses (D_in1,...,D_in6), control signals of multiplexers (sel_mux1,...,sel_mux6) and demultiplexers (sel_demux1,..., sel_demux6), synchronization input (clk), reset input (rst), and 6 output 8-bit buses (D_out1,...,D_out6).

Fig. 15 shows a diagram of the functional simulation of the combined sorting MOD by the method of «even-odd» permutation for 6 input data of 8 bits.

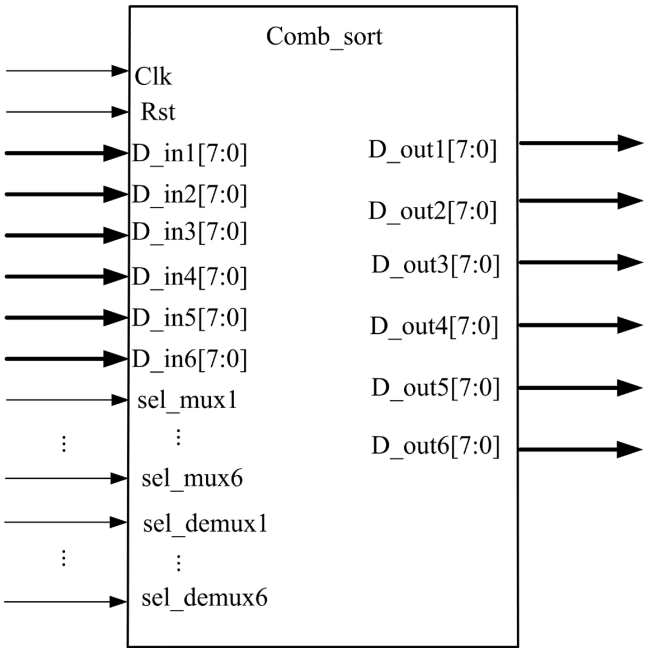

Fig. 14. Block diagram of a multi-clock sorting operating device of the combined type for $N=6$

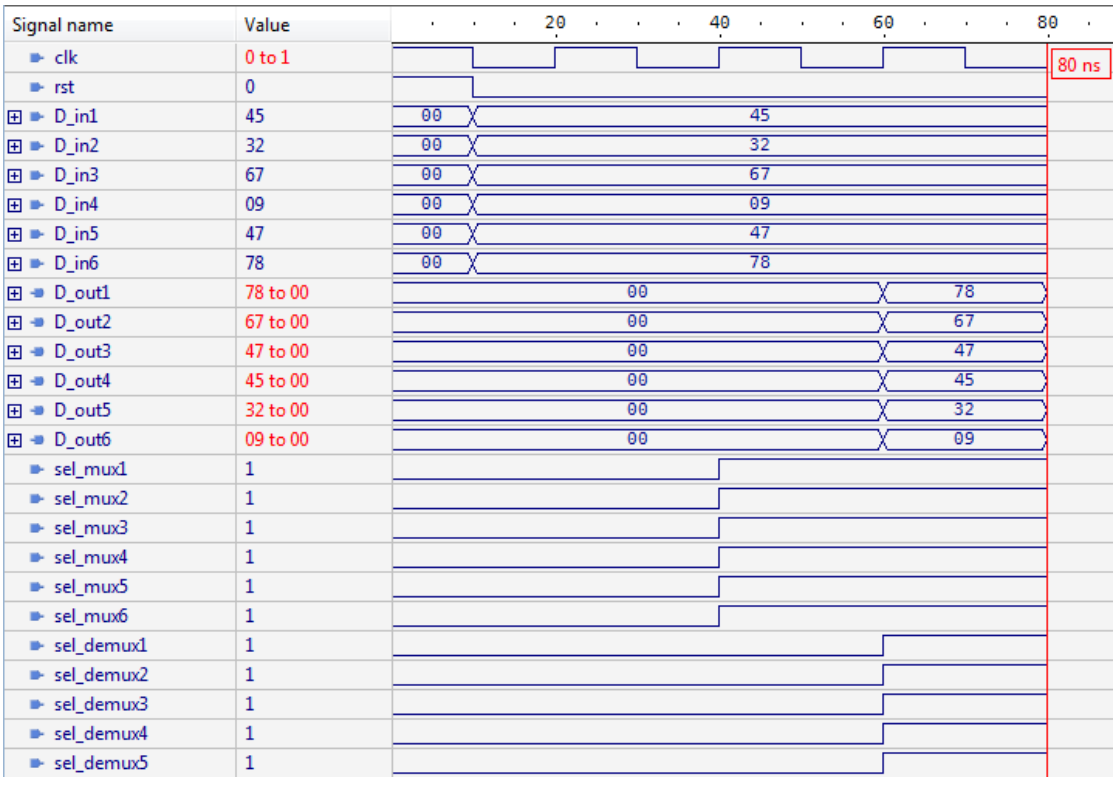

Fig. 15. Functional diagram of the operation of a multi-cycle sorting operating device of the combined type for $N=6$

In this diagram, we can see the input data and the generation of control signals for multiplexers and demultiplexers at each iteration. The final result is formed on the 4^{th} cycle (60 ns) since a MOD of the combined type at $N=6$ performs 3 iterations. In each iteration, 3 comparison and permutation operations are performed in parallel.

Table 1 gives the results of the synthesis of implemented sorting devices for 16 input single-byte numbers on the Xilinx FPGA of the Artix-7 family.

Table 1

Results of synthesis of sorting devices on FPGA

No.	Type of sorting device	Number of matching tables, (LUT)	Clock frequency, (MHz)
1	Algorithmic OD	704	332
2	Conveyor OD	1043	393
3	MOD of sequential type	452	302
4	MOD of sequential-iterative type	274	315
5	MOD of combined type	279	352
6	MOD of iterative type	217	54

As can be seen from Table 1, the greatest hardware costs are inherent in the conveyor OD and the smallest – in the MOD of the iterative type, which is confirmed by analytical calculations. The best ratio between hardware and time characteristics are demonstrated by MODs of sequential and combined types.

6. Discussion of results of investigating different types of binary sorting devices

On the basis of mathematical formulas (1) and (3), it is possible to calculate the hardware complexity of the algorithmic and pipeline ODs with any amount of input data (N). As a result of our research on the hardware complexity of the sorting OD data, with the use of an improved comparison scheme, a plot of dependence of the total number of gates M on the value of the input data N (Fig. 6) was built. It was established that the pipeline OD has 1.4 times greater hardware complexity than the algorithmic OD due to pipeline registers.

The use of an improved comparison scheme in the structure of the algorithmic OD made it possible to reduce its hardware complexity by 1.2 times and increase its speed by 1.8 times compared to known implementations [8, 9, 13, 16].

On the basis of mathematical formulas (2) and (4), it is possible to calculate the time complexity of algorithmic and pipeline ODs with any amount of input data (N).

As a result of our research into the time complexity of sorting OD data, with the use of an improved comparison scheme, a plot of dependence of the total number of microtacts τ on the value of the input data N was constructed (Fig. 7). Fig. 7 shows that the time complexity of the pipeline OD is 2 times less than that of the algorithmic OD. Conveyor OD has a minimum bandwidth, due to which it can be very effective in case of full filling of the conveyor during multi-threaded data processing.

The use of an improved comparison scheme in the structure of the pipeline OD made it possible to reduce its hardware complexity by 1.1 times and increase its speed by 1.6 times compared to known implementations [8, 9, 13, 16].

Analytical expressions for the calculation of hardware and time complexity were derived during the study of multi-

cycle ODs for sorting binary data, which allow determining the optimal ratio between these parameters for each MOD.

On the basis of mathematical formulas (6), (8), (10), (12), it is possible to calculate the hardware complexity of the corresponding MOD with any amount of input data (N).

Based on mathematical formulas (7), (9), (11), (13), it is possible to calculate the time complexity of the corresponding MOD with any number of input data (N).

The calculation of hardware complexity will allow one to determine which type of MOD requires the least amount of hardware resources, and which one requires the most. Calculation of the time complexity will make it possible to determine the speed of the corresponding MOD in microclocks. Determining these parameters is important when designing a corresponding MOD, when these parameters are set at the initial design stage.

As a result of our studies of the hardware complexity of the sorting MOD, a plot of dependence of the total number of gates M on the value of the input data N was built (Fig. 12). It was established that when the number of input data increases ($N \geq 128$), the hardware complexity of MOD of data sorting decreases by an order of magnitude in comparison with known implementations of single-cycle and conveyor ODs [8, 9]. This gives reason to claim that the use of such MODs has significant advantages and allows one to significantly save hardware resources and the area of the crystal when they are implemented on FPGA.

As a result of our research into the time complexity of sorting MOD, a plot of dependence of the total number of microtacts τ on the value of the input data N was constructed (Fig. 13). Fig. 13 demonstrates that the best time characteristics are inherent in the MOD of the combined type due to the incomplete merging of the vertices of the algorithm flow graph. The iterative MOD has the worst indicators of time complexity since all operations of the algorithm are performed sequentially in time due to one operation. This enables the designer to choose the appropriate MOD when solving the given task. For example, with non-critical time parameters, one can choose a slower type of device, and in the case of the importance of this parameter, combined options make it possible to significantly speed up the operation. The MOD of the combined type has 1.25 times worse performance compared to the performance of the conveyor OD.

When synthesizing the implemented VHDL models of sorting ODs on the Artix-7 FPGA family, the results of equipment costs and clock frequency were obtained (Table 1). From Table 1, it can be seen that conveyor and algorithmic ODs have the largest equipment costs, but their performance indicators are the highest. Among the different types of MODs sorting equipment, the iterative one has the lowest costs, and the sequential sorting device has the highest costs. The combined-type MOD has the best performance indicators. The optimal ratio between hardware and time characteristics during synthesis on a FPGA is demonstrated by MODs of sequential and combined types. With an increase in the amount of input data, the sorting MOD data approximately reaches a speed that is 2.3 and 3.4 times lower than that of known conveyor ODs, which is confirmed by theoretical calculations.

Developed operational data sorting devices can be used as components of specialized computer systems, arithmetic logic devices, and coprocessors. The use of algorithmic and pipeline sorting ODs makes it possible to speed up the data sorting operation by increasing the number of equipment. The use of multi-cycle ODs makes it possible to reduce the number of equipment with a slight decrease in speed for certain types of MODs, which leads to a decrease in the area of the FPGA crystal.

When designing a MOD for sorting binary data, certain research limitations are revealed, which are the complexity of designing such devices with a large amount of input data. Therefore, at the next stage of research, it is planned to develop an automated system for spatio-temporal transformation of the algorithm into the structure of the corresponding MOD. This will make it possible to significantly save the time of designing such devices and deepen the study of their system characteristics.

Prospects for the further development of our research are the development of methods for analyzing the structural complexity and functional completeness of multi-clock operating devices for sorting binary data. Reducing the number of MOD inputs and outputs and their intermediate signals could make it possible to reduce the area of the crystal on which such devices are implemented during their synthesis on FPGA.

At the same time, an important task is also the improvement of the scheme for comparing binary numbers and its components, which would make it possible to improve the system characteristics of various types of MODs for sorting binary data.

7. Conclusions

1. When researching different ways of hardware representation of the sorting algorithm by the method of «even-odd» permutation based on the algorithm flow graph, the algorithmic and conveyor devices for sorting binary data were built, and the formulas for calculating the equipment costs for their implementation were given. An improved scheme for comparing binary data is proposed, which, when applied in the structures of algorithmic and pipeline ODs, made it possible to reduce their hardware complexity by 1.2 times and increase their time complexity by 1.8 times.

2. As a result of our study of the hardware complexity of MODs built on the basis of STG, it was established that the combined MOD has the fewest logic gates among different types of MODs. When compared with known implementations of single-cycle and conveyor ODs for sorting a large amount of input data, the sorting MODs contain tens of times fewer logic gates.

The results of our study of the time complexity showed that multi-cycle ODs of the combined and sequential types have a time complexity of about 2.3 and 3.2 times less than that of the conveyor OD at large input values. The time complexity of the iterative MOD is 7.8 times less than that of the pipeline OD.

3. In the synthesis of OD for sorting on FPGA, an effective structure with an optimal ratio of hardware and time costs is a combined MOD. Compared to the conveyor OD, the combined MOD has 3.7 times lower hardware complexity and 1.1 times lower speed, which is characterized by a small amount of input data. The speed of the iterative MOD is 7.3 times lower than that of the pipeline OD, and the hardware complexity of the pipeline OD is 4.8 times greater than that of the iterative MOD. Thus, the results of practical implementation on FPGA mostly confirm the authenticity of theoretical calculations of system characteristics of various

types of sorting ODs. Having determined the necessary system characteristics, the designer can choose the optimal structure that has the best ratio between hardware and time characteristics for the implementation of the given task.

Conflicts of interest

The authors declare that they have no conflicts of interest in relation to the current study, including financial, personal, authorship, or any other, that could affect the study and the results reported in this paper.

Funding

The work was carried out within the framework of the project of the Ministry of Education of Ukraine «Elements of hybrid sensor microsystems for biomedical applications» (state registration number 0122U000858).

References

1. Knuth, D. E. (2011). The Art of Computer Programming, Volume 4A: Combinatorial Algorithms, Part 1. Addison-Wesley Professional, 912. Available at: https://ia804701.us.archive.org/2/items/B-001-001-251/B-001-001-251.pdf
2. Siewiorek, D., Robert, S. (2017). Reliable Computer Systems: Design and Evaluation. CRC Press, 908. doi: https://doi.org/10.1201/9781439863961
3. Dobre, C., Xhafa, F. (2013). Parallel Programming Paradigms and Frameworks in Big Data Era. International Journal of Parallel Programming, 42 (5), 710–738. doi: https://doi.org/10.1007/s10766-013-0272-7
4. Cormen, T. H., Leiserson, C. E., Rivest, R. L., Stein, C. (2009). Introduction to algorithms: third edition. Massachusetts Institute of Technology. Available at: https://pd.daffodilvarsity.edu.bd/course/material/book-430/pdf_content
5. Hanyu, T., Endoh, T., Suzuki, D., Koike, H., Ma, Y., Onizawa, N. et al. (2016). Standby-Power-Free Integrated Circuits Using MTJ-Based VLSI Computing. Proceedings of the IEEE, 104 (10), 1844–1863. doi: https://doi.org/10.1109/jproc.2016.2574939
6. Michailov, D. (2011). Hardware implementation of recursive sorting algorithms using tree-like structures and HFSM Models. Tallin: TUT Press, 114. Available at: https://digikogu.taltech.ee/en/Download/eb5d073c-02e5-46b9-83de-71fa17bd572a
7. Akl, S. G. (1985). Parallel sorting algorithms. Academic Press. doi: https://doi.org/10.1016/c2013-0-10281-4
8. Melnyk, A. O. (2008). Arkhitektura kompiutera. Lutsk: Volynska oblasna drukarnia, 470.
9. Melnyk, A. O. (2014). Pamiat iz vporiadkovanym dostupom. Lviv: Vydavnytstvo NU «Lvivska politekhnika», 296.
10. Yakovlieva, I. D. (2008). Otsinka variantiv syntezu paralelnykh obchysliuvalnykh prystroiv sortuvannia. Visnyk Natsionalnoho universytetu «Lvivska politekhnika», 630, 124–130. Available at: https://ena.lpnu.ua/handle/ntb/880
11. Tsmots, I. G., Antoniv, V. Ya. (2016). Algorithms and Parallel Structures for Data Sorting Using Insertion Method. Scientific Bulletin of UNFU, 26 (1), 340–350. doi: https://doi.org/10.15421/40260153
12. Tsmots, I. G., Antoniv, V. Y. (2020). Improvement of parallel sorting by method of merging. Scientific Bulletin of UNFU, 30 (4), 134–142. doi: https://doi.org/10.36930/40300422
13. Gryga, V., Nykolaichuk, Y., Vozna, N., Krulikovskyi, B. (2017). Synthesis of a microelectronic structure of a specialized processor for sorting an array of binary numbers. 2017 XIIIth International Conference on Perspective Technologies and Methods in MEMS Design (MEMSTECH). doi: https://doi.org/10.1109/memstech.2017.7937560
14. Dunets, R., Gryga, V. (2015). Spatio-temporal synthesis of transformation matrix of reverse fast cosine transformation. The Experience of Designing and Application of CAD Systems in Microelectronics. doi: https://doi.org/10.1109/cadsm.2015.7230792
15. Gryga, V., Kolosov, I., Danyluk, O. (2016). The development of a fast iterative algorithm structure of cosine transform. 2016 13th International Conference on Modern Problems of Radio Engineering, Telecommunications and Computer Science (TCSET). doi: https://doi.org/10.1109/tcset.2016.7452100
16. Gryga, V. (2018). Design and research of operational and pipelined binary number sorting devices. 18th International Multidisciplinary Scientific GeoConference SGEM2018, Informatics, Geoinformatics and Remote Sensing. doi: https://doi.org/10.5593/sgem2018/2.1/s07.036
17. Alotaibi, A., Almutairi, A., Kurdi, H. (2020). OneByOne (OBO): A Fast Sorting Algorithm. Procedia Computer Science, 175, 270–277. doi: https://doi.org/10.1016/j.procs.2020.07.040
18. Alaparthi, S., Gulati, K., Khatri, S. P. (2009). Sorting binary numbers in hardware - A novel algorithm and its implementation. 2009 IEEE International Symposium on Circuits and Systems. doi: https://doi.org/10.1109/iscas.2009.5118240
19. Kobayashi, R., Miura, K., Fujita, N., Boku, T., Amagasa, T. (2022). An Open-source FPGA Library for Data Sorting. Journal of Information Processing, 30, 766–777. doi: https://doi.org/10.2197/ipsjjip.30.766
20. Mihhailov, D., Sklyarov, V., Skliarova, I., Sudnitson, A. (2011). Hardware implementation of recursive sorting algorithms. 2011 International Conference on Electronic Devices, Systems and Applications (ICEDSA). doi: https://doi.org/10.1109/icedsa.2011.5959040
21. Deo, N. (1974). Graph theory with applications to engineering and computer science. Dover Publications. Available at: https://www.shahucollegelatur.org.in/Department/Studymaterial/sci/it/BCS/FY/book.pdf

22. Kogut, I. T., Druzhinin, A. A., Holota, V. I. (2011). 3D SOI Elements for System-on-Chip Applications. Advanced Materials Research, 276, 137–144. doi: https://doi.org/10.4028/www.scientific.net/amr.276.137
23. Novosiadlyi, S., Kotyk, M., Dzundza, B., Gryga, V., Novosiadlyi, S., Mandzyuk, V. (2017). Formation of carbon films as the subgate dielectric of GaAs microcircuits on Si-substrates. Eastern-European Journal of Enterprise Technologies, 5 (5 (89)), 26–34. doi: https://doi.org/10.15587/1729-4061.2017.112289
24. Kehret, O., Walz, A., Sikora, A. (2016). Integration of hardware security modules into a deeply embedded tls stack. International Journal of Computing, 15 (1), 22–30. doi: https://doi.org/10.47839/ijc.15.1.827
25. Gryga, V., Dzundza, B., Dadiak, I., Nykolaichuk, Y. (2018). Research and implementation of hardware algorithms for multiplying binary numbers. 2018 14th International Conference on Advanced Trends in Radioelecrtronics, Telecommunications and Computer Engineering (TCSET). doi: https://doi.org/10.1109/tcset.2018.8336427
26. Giachetti, R. (2016). Design of enterprise systems: Theory, architecture, and methods. CRC Press. doi: https://doi.org/10.1201/9781439882894
27. Hryha, V. M., Nykolaichuk, Ya. M., Hryha, L. P. (2022). Pat. No. 151889. Prystriy porivniannia bahatorozriadnykh dviykovykh danykh. No. u202200478; declareted: 07.02.2022, published: 28.09.2022. Available at: https://sis.ukrpatent.org/uk/search/detail/1707803/