PARALLEL ALGORITHMS AND STRUCTURES FOR IMPLEMENTATION OF MERGE SORT

Ivan Tsmots, Oleksa Skorokhoda, Volodymyr Antoniv
Lviv Polytechnic National University, Ukraine, Lviv, 79000

Requirements for development of real-time hardware have been formed and principles of their building have been selected. Consistent flow graphs for algorithms of merge sort of data sets have been developed. Hardware for data sorting with high-efficiency of equipment use has been synthesized on their basis.

Keywords: data sorting, merge sort, hardware, flow graph, real-time.

I. INTODACTION

The development of information technologies is characterized by the expansion of applications, many of which requires parallel sorting of data sets in real-time. You can provide such data sorting by using of specialized tools, which architecture maps the structure of the sorting algorithm on hardware and is oriented on VLSI implementation. Implementation of highly efficient specialized sorting tools requires extensive use of modern element base, the development of new methods, algorithms, and VLSI structures. Real-time and VLSI implementation of sorting algorithms with high-efficiency of equipment use are provided by parallelization and pipelining of sorting processes, hardware mapping of algorithms' structures on the architecture, which is adapted to the intensity of the data streams flow. The orientation of sorting tools structures on VLSI implementation requires reducing of the interface pins number and implementation of algorithms based on the same type of processor elements (PE) with regular and local connections.

Therefore the problem of developing of new effective methods and algorithms for data sets sorting, which are focused on adapting to the intensity of the data streams flow and VLSI implementation, is particularly relevant.

II. ANALYSIS OF PUBLICATIONS

Analysis of methods for parallel sorting of data sets [1-13] shows that for the hardware implementation most suitable methods are counting sort, insertion sort and merge sort. Comparison of these methods [8-10] shows that parallel implementations of merge sort in comparison with other algorithms are more structured, homogeneous and oriented on parallel-pipelined implementation using VLSI. At the core of merge sorting algorithms is basic operation for combining of two or more arranged arrays into one ordered array. Hardware implementation of basic operations for combining three or more arranged arrays into one ordered array is complex and requires significant hardware costs. More simple is basic operation for combining two ordered arrays into one ordered arrays, i.e. two way merging. The disadvantage of existing algorithms for implementation of two way merging is a small performance because they are based on the operations of pairwise comparison of data items [8,15]. One of the ways to improve execution performance of two way merging is its implementation based on the basic operation of multi-channel merging and sending of data groups. Number of concurrent basic operations determines the performance of means for data sorting, that due to the symmetry of most algorithms for numbers sorting may be different. Depending on the structure of the implemented algorithm and the requirements of a particular application various structures of sorting device can be synthesized, that are different by organization of sorting process and technical parameters [5].

An analysis of publications [1-16] shows that synthesis of highly efficient hardware for real time data sorting using merge sorting algorithm requires the development of new coordinated-parallel algorithms and methods of spatiotemporal converting them to parallel structures.

III. FORMATION OF GOALS OF THE ARTICLE

The aim of this work is to develop requirements and to select principles of building real-time hardware, to develop consistent flow graphs for algorithms of merge sort of data sets and to synthesize on their basis hardware with high efficiency of equipment use.

IV. MAIN PART

Formation of requirements and selection of principles of building hardware for data sets sorting. One of the most widely common requirement relates to the tools for parallel sorting of data sets is to provide high performance. This problem occurs usually when using such tools for sorting intensive streams of data sets in real time. To provide real-time processing the time of sorting T_s should not exceed the time of data incoming T_{fd}, ie:

ISSN: 2278 – 1323

$$T_s \le T_{fd}$$

In addition, these tools should have high efficiency of equipment use, that takes into account number of interface outputs, links performance with equipment costs and assesses elements (valves) productivity. The quantitative value of efficiency of equipment use E of algorithms for sorting data at VLSI implementation is defined as

$$E = \frac{R}{t_s(\sum_{i=1}^{N} W_{PE_i} + k_1 Y + k_2 P)}$$

where R - complexity of algorithms for sorting data that is determined by the number of pair wise comparison operations; W_{pe} - the cost of equipment for the implementation of PU; N – the number of PU; ts - time spent on sorting of data set; Y - number of interface outputs; k_1 – factor taking into account the number of interface outputs $k_1=f(Y)$; P - the number of connections between PU; k_2 – factor taking into account the number of connections between PU $k_2=f(P)$.

One of the conditions to achieve high efficiency of equipment use when sorting data sets in real time is the fulfillment of such conditions:

$$P_d \le D_c ,$$

where $P_d=kn_kF_d$ - the intensity of incoming data flow; $D_c = \frac{sn_s}{T_к}$ - the intensity of sorting; k – number of channels of incoming data flow; n_k - digit capacity of input data channels; F_d – frequency of input data flow; s – the number of data sorting channels; n_s – digit capacity of data sorting channels; $T_к$ - conveyor clock cycle of the sorting device.

For VLSI implementations sorting algorithms should be well structured, focused on implementation on the set of interconnected PU and ensure deterministic data migration. Structure and operations, performing by PE, depends on the requirements that apply to the time of sorting. When developing sorting algorithms for VLSI implementations it is necessary to simultaneously consider many interrelated factors. First of all sorting algorithms should be recursive and locally dependent. In recursive algorithm all PU should perform the same operation.

Data sorting algorithms can be divided into two classes: with local and global connections between PU. To assess the connections between PU, temporal and hardware complexity of algorithm realization the grid model should be developed. This model consists of a set of PU that form point systems (grids) in Cartesian coordinate system [5]. In grid model, for each PE time j and spatial *i* indexes are assigned, that indicate when and where each of operations are executed. In algorithms with local sending data between the spatial indexes *j* in step recursion is limited by some constant, because in such algorithms exchanges are carried out only between nearest adjacent PU. Algorithms that at recursion have spaced spatial indexes belong to class of algorithms with global connections. The complexity of implementation of exchanges between PE depends on digit capacity of data transmission channels, and on the distance between them, which is defined as difference of two spatial indexes.

Cost of VLSI for parallel sorting of data sets mainly depends on the area of the crystal, which is defined as cost of equipment (number of transistors) and the number of external pins, which number are restricted to the level of technology and the crystal size. Orientation of structures for data sorting on VLSI-implementation requires reducing the number of interface outputs and the number of connections between the PU.

Development of highly efficient tools for parallel sorting of data sets can be provided with an integrated approach that includes:

- research and development of methods and algorithms for parallel sorting of large data sets;
- development of new structural and circuit solutions focused on VLSI technology;
- means for computer-aided VLSI-design, which provide reduction of terms and increase of the design quality.

To ensure high efficiency of equipment use of VLSI-structure for sorting data sets in real time it is proposed to use the following principles [5,6]:

- parallelization of data sorting process ;
- specialization and adaptation of hardware to structure of sorting algorithms and intensity of income data flow;
- homogeneity of PU and regularity of relations between them;
- coherence of sorting intensity with intensity of income data flow.

Forms of sorting algorithms reflection. To assess the computational and structural characteristics of sorting algorithms their representation in the form of functional graph $F=(P,G)$ is used, where $P=\{P_1,P_2,...,P_n\}$ – a set of functional operators, G- law of reflection of connections between operators [5]. Functional graph of algorithm reflects the sequence and interdependence of functional operators. Graphically functional graph of algorithm appears as node of graphs that correspond to operators of algorithm F_i and edges that reflect the relationships between operators. The complexity of functional operators F_i is defined by structural units of information and complexity of performed

ISSN: 2278 – 1323

operations. Such representation of algorithm does not fully reflect the time-spatial relationship between functional operators.

To detect parallelism of data sorting algorithm and to manage it for finding the optimal space-time decisions, functional graph should be presented in tiered-parallel form (TPF) [5].

In this form of algorithm representation all functional operators Fi are distributed on tiers so on jth tier there are functional operators that depend at least from one functional operator on j-1th tier and do not depend from functional operators on following tiers. Within the tier functional operators do not have connections between themselves.

Each j-th tier of algorithm is described by the following parameters:

- sets of independent functional operators F_{ij}, where j – number of a tier, i – number of a functional operator on the tier.
- set of data input channels and issuing of interim results;
- digit capacity of each communication channel;

The number of tiers in TPF of algorithm is its height h, and the maximum number of functional operators on the tier determines the width L. Functional operators are placed using space-time indexes in the "time-space" coordinate system. Such reflection of a oriented graph of algorithm we will call "flow-parallel form" or flow graph [5, 6]. Such parameters of flow graph as the complexity of functional operators F_{ji}, width L and height h are mutually dependent, so changing one of them leads to changes of others.

To move from a functional graph to its reflection in form of flow graph it is necessary to write him in a form of a matrix n × n, where "1" corresponds to the availability of the communication channel, "0" – to the communication absence [5]. The matrix is formed in such a way that for each data source functional, operator F_{ji} a line is formed, that reflects its connections with other functional operators. If we denote the columns of the matrix by vectors, we can determine the resulting column vector:

$$\vec{V}_0 = \vec{V}_{F1} + \vec{V}_{F2} + ... + \vec{V}_{Fn}$$

In vector we determine the number of zero elements , for example, the second and the fifth. According to predefined numbers we find functional operators that does not have descendants and therefore forms a zero tier, in this case F_2 and F_5. Then by the formula we calculate:

$$\vec{V}_1 = \vec{V}_0 - \vec{V}_{F2} - \vec{V}_{F5}$$

In the vector we find indexes of zero elements, by which we define functional operators that form the first tier. Similarly we calculate following vectors and determine the functional operators that form following tiers.

To move from sorting algorithms to VLSI architectures with high efficiency of equipment use we propose to reflect these algorithms in a form of consistent flow graph.

Developing of a consistent flow graph for sorting algorithms. The process of development of a consistent flow graph for sorting algorithms can be divided into the following four stages:

1. Decomposition of the problem solving algorithm.
2. Design of communications (data exchange) between the functional operators.
3. Consolidation of the functional operators.
4. Planning of sorting process.

At the stage of decomposition an algorithm for sorting of data sets is divided into functional operators Fji between which connections corresponding to this sorting algorithm are established. The greater the degree of detailing of algorithm we get as a result of decomposition, the easier we can adapt it to the requirements of a particular application. Decomposition can be carried out by the method of functional decomposition. Using of functional decomposition provides opportunity to get space-time mapping of sorting algorithm structure based on elementary operations of pairwise comparison and rearrangement of data. Time and method of execution of these operations by PU is one of the key parameters in determining the conveyor tact T_k and the digit capacity of data channels n_k in VLSI structures for data sorting. The result of the first stage of development is graph scheme of the algorithm where functional operators F_{ji} have approximately the same time of execution, and their complexity is determined by the required performance.

At the stage of communications design it is necessary to determine a structure and digit capacity of data exchange channels between functional operators F_{ji}. For this purpose we carry out the transition from the graph scheme of algorithm to flow graph in which the space-time placement and fixing of functional operators F_i on tiers is carried out. Structure of connection in flow graph between functional operators F_{ji} of adjacent tiers is defined by number of input data channels and their digit capacities. Example of flow graph of parallel merge data sorting algorithm for array of 16 numbers is shown in Fig.1, where F_{ji} – functional operator for pairwise comparison and rearrangement of data.

ISSN: 2278 – 1323

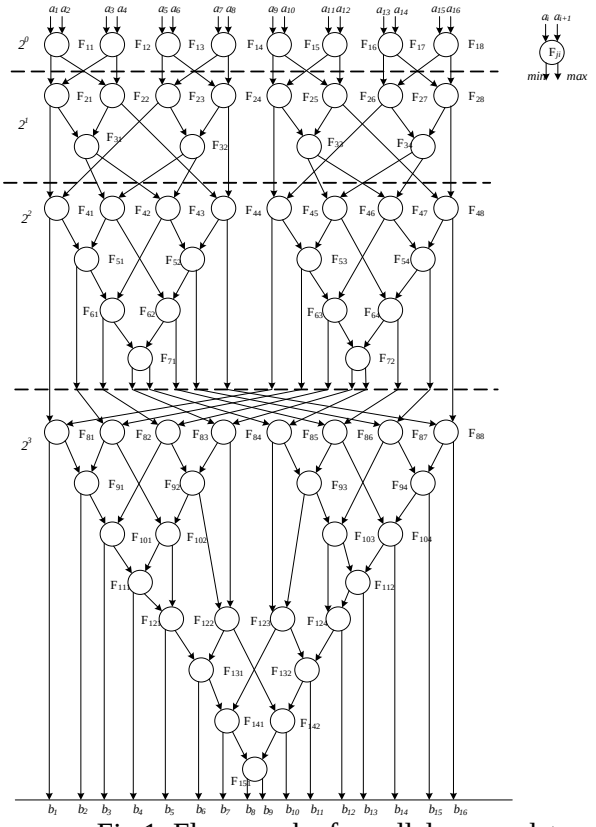


Fig 1. Flow graph of parallel merge data sorting algorithm for array of 16 numbers

At the core of sorting algorithms by merging is basic operation of union of two ordered arrays $\{a_{1i}\}_{i=1}^{2^{i-1}}$ and $\{a_{2i}\}_{i=1}^{2^{i-1}}$ into one ordered array. At the beginning of sorting the input array of numbers $\{a_j\}_{j=1}^{N}$ is divided on N/2 ordered arrays with length of one. As a result of the first basic operation N/4 ordered arrays with length of two are formed. Number of different types of basic operations, needed for sorting of array of N numbers is determined by the formula.

$$k=\log_2 N.$$

The aalgorithm for merge sorting of data sets is implemented on the basis of this functional operator F_{ji}:

$$y=\begin{cases}0, when\ a_1 \le a_2\\ 1, when\ a > a_2\end{cases},$$

$$b_1=\begin{cases}a_1, when\ y=1\\ a_2, when\ y=0\end{cases},\quad b_2=\begin{cases}a_1, when\ y=0\\ a_2, when\ y=1\end{cases},$$

where y – the result of comparing of two numbers; a_1, a_2 – numbers to compare; b_1, b_2 – outputs of bigger and smaller sorted numbers.

The flow graph of parallel merge data sorting algorithm for array of 16 numbers has a height h = 15, width L = 8 and is realized on the basis of 4 types of basic operations. On the first tier of the given graph basic operations of first type are performed, on the second and third tiers – basic operations of second type, on the 4th-7th tiers – basic operations of third type, and on the 8th-15th tiers – basic operations of fourth type

According to the results of the first two stages of development we can estimate an intensity D_C of data set sorting, which can be achieved by hardware implementation of flow graph for sorting. Initial data for determining the intensity of data sets sorting $D_C=\frac{sn_S}{T_K}$ are:

- number of channel for data sorting s and their digit capacities ns;
- complexity of functional operators F_{ji}, which determines the time for their implementation and the conveyor tact of sorting T_c;
- performance of element base, which is considered when determining the conveyor tact of sorting T_c.

To assess the consistency of data flow intensity P_d with computing ability D_C we introduce the consistency coefficient, which is defined as:

$$L=\left\lceil\frac{P_d}{D_c}\right\rceil$$

where $\lceil\ \rceil$ - rounding to the upper integer.

Consistency factor L can be $L = 1$, $L > 1$ and $L < 1$. When L=1, then developed graph for data sorting is consistent and its hardware implementation provides high efficiency of equipment use.

In case when L>1 developed graph for data sorting is not consistent. To improve its consistency we should increase the intensity of sorting D_C. Increasing of the intensity of sorting D_C can be achieved by increasing the number of channel for data sorting s and their digit capacities ns, or by decreasing the complexity of functional operators F_{ji} In case when the changing of these parameters doesn't help to reach the necessary intensity of data sorting Ds, then it can be achieved by parallel implementation of L graphs.

In *the third stage of design* the consolidation of operations are made, by combining functional operators F_{jk} and data transmission channel within a tier and between tiers. This stage is used for the case when L <1, ie when you need to reduce the intensity of sorting D_C. The sort graph that we get as a result of combining we will call specified flow graph. The stage of consolidation is closely linked with the stage of planning of sorting process.

The fourth stage of the planning of sorting process comes down to preservation of information about the structure of the flow graph of sorting algorithm. At this stage the planning of sorting process is executed and values of delays and data reshuffle are determined. To map the

ISSN: 2278 – 1323

data sorting process into the specified flow graph operators for management, delays and the reshuffle of data are introduced. Consider the three main options for operations consolidation (combining of functional operators) for obtaining a specified flow graph.

The first variant of obtaining a specified flow graph of data sorting is its linear projection on the horizontal axis X. In this case consolidation of operations is carried out by combination of functional operators and data transfer channels between tiers. Example of projection of parallel merge data sorting algorithm for array of 16 numbers on the horizontal axis X is shown in Fig. 2, where F_{ji} – functional operator of pairwise comparison and rearrangement of data, F_{MDS} – macro operator of delay and shuffle, F_{MM} – control macro operator.

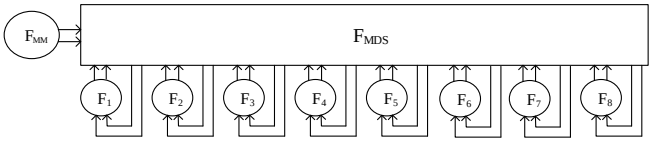


Fig. 2. Linear projection of graph for parallel merge data sorting algorithm for array of 16 numbers on the horizontal axis X

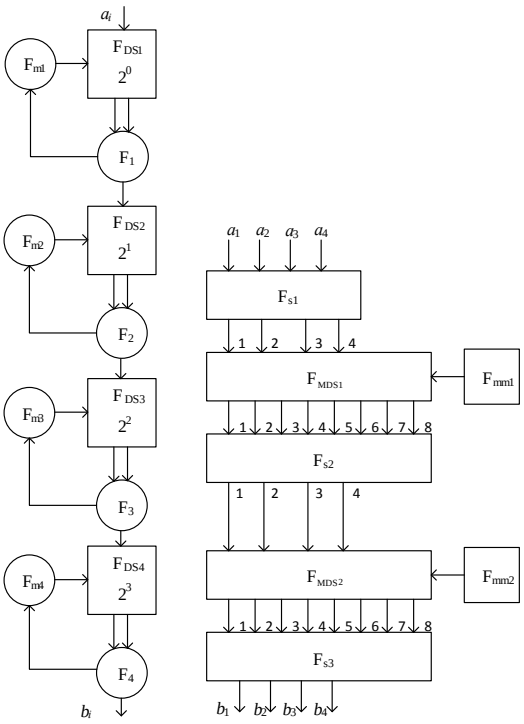
The estimated intensity of data sets sorting for a linear projection on the horizontal axis is

$$D_c = \frac{sn_s}{(N-1)T_к},$$

where N – count of numbers for sorting.

The second option to obtain a specified flow graph of merge data sorting is its linear projection on the vertical axis Y. In this case, consolidation of operations is carried out by combination of functional operators and data transfer channels both within a tier and between tiers. As a result of such consolidation we will have a specified flow graph with one data transfer channel and four types of basic operations for merging of two ordered arrays of length 2^{i-1} into one ordered arrays of length 2^i, where i=1,…,N.

Example of projection of parallel merge data sorting algorithm for array of 16 numbers on the vertical axis Y on the basis of single-channel two-way merging is shown in Fig. 3a, where F_{DS1} – F_{DS4} – operators of delay and shuffle for single-channel two-way merging; F_{M1} - F_{M4} – control operators.



а) *б*)

Fig.3. Linear projection on the vertical axis Y of the graph of sorting algorithm for array of 16 numbers
а) on the basis of single-channel two-way merging;
б) on the basis of four-channel two-way merging.

The estimated intensity of data sets sorting for a linear projection on the vertical axis Y is

$$D_c = \frac{n_s}{T_к}$$

The third option to obtain a specified flow graph of merge data sorting is to use for its implementation basis operations of multi-channel two-way merging. The number of channels for two-way merging depends on the consistency factor L. The required number of basic operations of two-way parallel merging for sorting an array with N numbers is determined by the formula:

$$g = \left\lceil \log_2 \frac{N}{m} \right\rceil$$

where m – number of channels.

Projection on the vertical axis Y of the sorting graph on the basis of multi-channel two-way merging is shown in Fig. 3B, where F_{MM} – control macro operator; F_{s1} – functional operator of sorting of four numbers; F_{s2}, F_{s3}- functional operators of sorting of eight numbers.

The estimated intensity of data sets sorting on the basis of multi-channel two-way merging for linear projection on the vertical axis Y equal:

$$D_c = \frac{mn_s}{T_к}.$$

ISSN: 2278 – 1323

The main way to increase the intensity of data sets sorting on the basis of multi-channel two-way merging is to increase the number of channels.

The main increase in intensity by sorting data sets based on multi merger is to increase the number of channels.

Synthesis of hardware for data sets sorting in real time. For the synthesis of hardware for data sets sorting in real time we will use known method of adequately hardware mapping of algorithm graph structure.

When using this method for every functional operator a PU or a sorting unit is assigned, for macro operator of delay and shuffle – memory and switches, for control macro operator – control unit, and for the arc between functional operators – data transfer channels [5]. Hardware synthesized in this way is algorithmic. In such structures an algorithm is implemented when passing and processing from input to output across all operating units. By the operating modes algorithmic structures are divided into synchronous and asynchronous.

In asynchronous (single-cycled) structures data processing is carried out without intermediate-remembering. Each single-cycle structure is sequential in terms of implementation of functional operators F_{ji}. It is the cause of speed limit and inefficient use of equipment when processing intensive data streams in real time. Therefore, for processing of data streams it is appropriate to use synchronous structures with conveyor implementation of algorithm graphs, in which execution of functional operators of an algorithm on different data are combined in time. Pipelining of algorithmic structures involves separating them on the steps by the addition of buffer memory. In this case, each step of the conveyor consists of two components: a buffer memory and a PU (sorting unit) of control unit which implements operators on the tier.

To ensure high performance and efficiency of functional equipment operators, implemented in steps conveyor should be simple and have approximately the same time of implementation. To ensure high performance and efficiency of equipment use functional operators that are implemented in step of conveyor should be simple and have approximately the same time of execution. Single-cycle algorithmic devices can be viewed as a one-step pipeline. So it is important to consider issues related to the synthesis of real-time conveyor algorithmic structures with high efficiency of equipment use.

Input information for the synthesis of devices for data sets sorting in real time are:

- number of input data N;
- digit capacity of input data;
- interface requirements;
- intensity of input data flow $P_d=kn_kF_d$;
- technical and economical requirements and restrictions.

The process of synthesis of hardware for real-time data sets sorting is multilevel and iterative, with the analysis of the characteristics, rollbacks and reviewing of previous decisions. Sequential functional decomposition of an algorithm of data sets sorting with definition of the hardware interface, functions of each functional operator, control-, delay- and reshuffle operators reflects the synthesis process "from top to down".

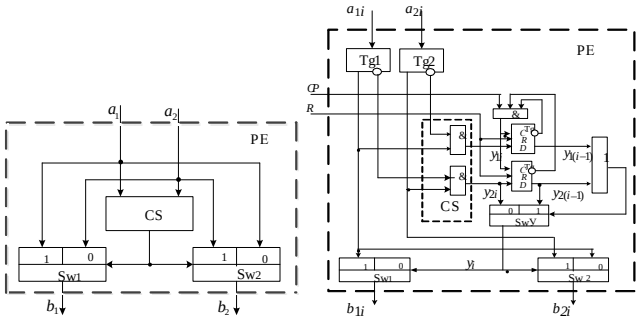
In the synthesis of real-time data sets sorting devices to achieve high efficiency of equipment use it is necessary to provide:

- consistency of data flow intensity with sorting intensity;
- minimizing of hardware costs to the level when still providing real-time.

The main ways to minimize hardware cost and to provide consistency of data flow intensity with sorting intensity are:

- selection of effective methods and algorithms of data sort;
- selection of functional operators complexity;
- changing of input data channels digit capacity and PUs digit capacity;
- changing of input data channels quantity.

In the synthesis of matrix device for parallel merge sorting for each functional operator a PU is assigned, that are interconnected by data transfer channels according to the flow graph of parallel sorting algorithm (Fig. 1). Structure of possible PUs that implement the functional operator,is shown in Fig. 4, where Tg - trigger km - switch, CS – comparison scheme , CI - clock input, R – reset to zero input.



a) *b*)

Fig.4. Structure of PU:
a) with parallel arrival of all data bits
b) with bitwise arrival of data

In a matrix device for parallel merge sorting one of the options to provide consistency of data flow intensity with sorting intensity is changing of

ISSN: 2278 – 1323

data input channels digit capacity and PUs digit capacity. On the Fig.4. two types of PU are shown:

- with parallel arrival and comparison of data (fig. 4a);
- with bitwise arrival and comparison of data (fig. 4b).

These two types of PU is extreme options of using vertical-group data arrival and comparison. In PU in fig. 4a numbers arrives in parallel code and their comparison performs in a single cycle. In PU in fig. 4b numbers arrives bitwise and their comparison performs in n cycles. Formation of the result of bitwise comparison of y_{1i} and y_{2i} in PU is carry out by comparison scheme, which is implemented on two AND gates. The result of previous comparison stores in triggers. Writing into trigger can be blocked by logic 0 signal from its inverse output. Before sortingof new data set triggers of results of comparison are set to zero. In other types of PU numbers arrives and compares in groups, digit capacity of which is 1<p<n.

The second option to provide consistency of data flow intensity with sorting intensity is using of hybrid algorithms, that are based on combining of merging and counting sort.

A second option coordination intensity data flow intensity is of sorting hybrid algorithm based on combining techniques merge sort and count. With the implementation of the hybrid algorithm to sort data carried out by fusion to PE that perform of sorting of numbers by counting. In a hybrid algorithm sorting of data is carry out by merge sorting on PU, that perform sorting of data set by counting method.

Devices that implement hardware projection of flow graph of algorithm for parallel merge data sorting on the horizontal and the vertical axes are slower. The structure of device that implements hardware projection of flow graph of parallel merge data sorting algorithm for array of 16 numbers on the horizontal axis X is shown in Fig. 5, where MFIFO – FIFO memory with programmable delay, CU – control unit.

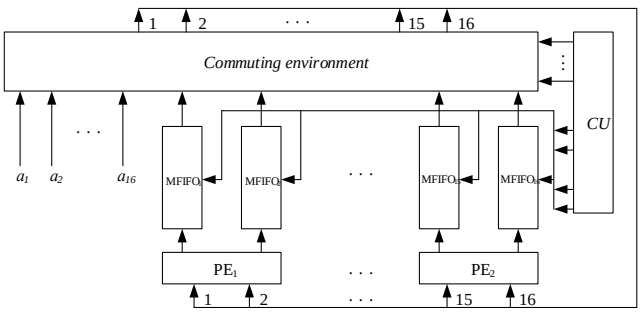


Fig. 5. Structure of recursive device for data sets sorting

The feature of the structure of this device for data sorting is system of feedback connections, which makes it recursive. The basis of recursive data sorting device is set of PU, whose structure is shown in Fig. 4a. Sorting in this device is going in cycles, number of which is determined by number of tiers in flow graph of algorithm. In each j_{th} work cycle of recursive data sorting device functional operators F_{ji} of jth tier are performed. To ensure the spatiotemporal deployment of a data sorting flow graph control units, MFIFO and switching environment are used. They provide in accordance: formation of control signals, delay of data on the required number of cycles and switching of data transfer channels according to data sorting flow graph [15]. Sorting of an array of 16 numbers in a recursive device is performed in 15 cycles. A recursive data sorting device is advisable to use to sort single data sets.

To sort intensive data streams it is advisable to use flow structures, that is a hardware projection of flow graph of parallel merge data sorting algorithm on the vertical axis Y (fig. 3). Flow structures of sorting devices based on two-way merging are implemented on the basis of PUs, which differ only by the volume of specialized memory. The structure of l-th (where l = 1,...k) PU for single-channel flow data sorting device based on two-way merging is shown in Fig. 6, where Rg – register, Sw – switcher, CU – control unit.

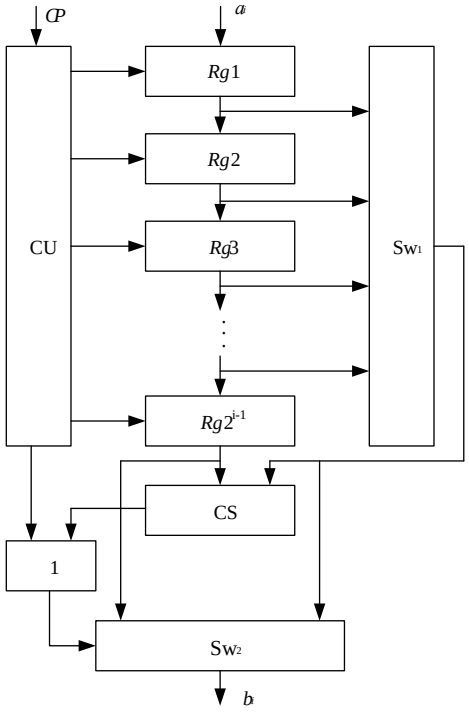


Fig. 6. Structure of PU for single-channel flow data sorting device

For the synthesis of single-channel flow data sorting device of N numbers $k=\log_2 N$ PU is

ISSN: 2278 – 1323

needed [14]. Each l-th PU includes specialized memory that consists of 2^{l-1} registers and $(2^{l-1}-1)$-input switch Sw_1. In l-th PU from two arrays of length 2^{l-1} in the output of Sw_2 an ordered array of length 2^l is formed. Managing of the data sorting process in each l-th PU is performed by CU, that remembers information about all performed comparisons and on its basis forms control signals for Sw1 and Sw2. Single channel flow data sorting device works on conveyor principle with a cycle equals $T_к=t_{Rg}+t_{CS}+t_{CU}+2t_{Sw}$, where t_{Rg}, t_{CS}, t_{CU} і t_{Sw} – accordingly delay time of a register, of a control unit, of a comparison scheme and of a switch. To perform sorting of array of N numbers it should complete $\sum_{l=1}^{k} 2^l$ cycles.

Reducing of data sorting time can be achieved by increasing of number of two-way merging channels. To implement this approach multi-channel flow data sorting device should be synthesized, which reflects in hardware projection of a flow parallel merge data sorting device on the vertical axis Y (fig. 3b). Such device is synthesized on the basis of PU, which consists of the following components: specialized parallel memory (SPM) and unit of parallel merging of two groups of data elements (UPM) (see Fig. 7). The number of PU required for the synthesis of multi-channel flow data sorting device is determined by the formula $g = \left\lceil \log_2 \frac{N}{m} \right\rceil$.

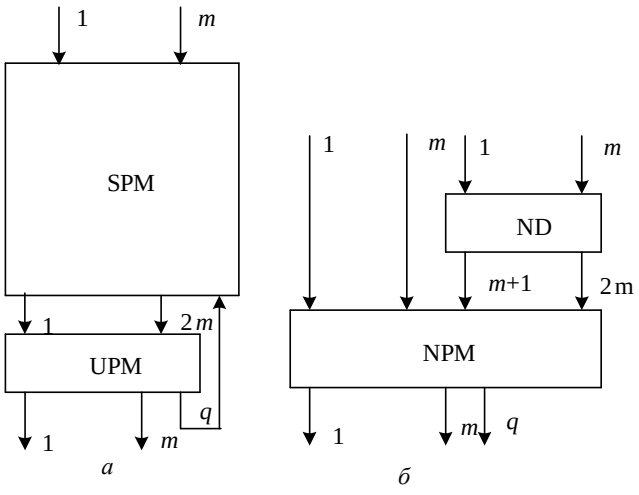


Fig. 7. Structures of a: a) processing unit, b) parallel merging unit.

To provide real-time sorting of data streams these components must meet the following requirements:

- SPM – to perform storage, concurrent arriving of m and issuing of 2m data elements per one parallel merging node, where the first m – ordered elements of the array $\{a_i\}_{i=1}^{m2^{p-1}}$, and the second m – ordered elements of the array $\{b_i\}_{i=1}^{m2^{p-1}}$, where (p=1,…,g);
- UPM – to perform fast single-cycle data elements merging of two ordered arrays $\{a_h\}_{h=1}^{m}$ and $\{b_h\}_{h=1}^{m}$ into one ordered array $\{c_h\}_{h=1}^{2m}$ where c_m -th element is accompanied by information on the input number from which it came.

Parallel merging of two data groups $\{a_h\}_{h=1}^{m}$ and $\{b_h\}_{h=1}^{m}$ that comes from the outputs of SPM, performs by parallel merging unit (PMU), whose scheme is shown in fig. 7b, where SN – shift node, PMN – parallel merging node. For efficient parallel merging of data groups at first you should order data, that arrives from (m+1)-th, …, 2m-th outputs. To provide high performance we propose to perform this function on the switches. In this case ordering time performed by PMN will be determined by time delay of passing information through the switch. To implement the PMN you should select fast parallel merge algorithm for ordered data groups. The analysis of methods and algorithms for parallel data sorting showed [1, 14], that the fastest hardware implementation of single-cycle parallel merging of two ordered groups is performed by the counting method. Time of this merging is determined by the formula:

$$t_{PM} = t_{PC} + t_A + t_D + t_{Sw},$$

where t_{PC} – time of pairwise comparison; t_A – adding time; t_D – decryption time; t_{Sw} – delay time on the switch.

The structure of SPM for *p-th* PU is shown in Fig. 8, where *k = 1, ..., Q / m*, ME - memory environment; MU - memory unit; MC – memory cell; SE - switching environment; SN - switching node; Sw - switch; MMU - memory management unit.

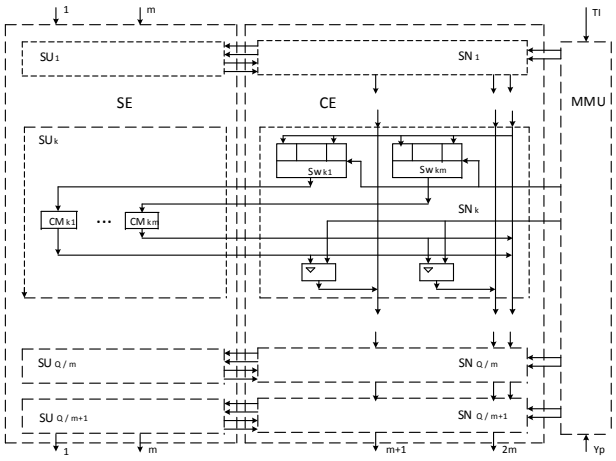


Fig. 8. The scheme of specialized parallel memory

ISSN: 2278 – 1323

The peculiarity of SPM is serial-parallel organization. Serial connection between MCs is implemented by using (m+1)-input switches Sw, that can transmit an information without shift or with shift from one to m data elements [5,16]. Parallel connection of MC columns of memory environment, which is provided by means of buffered amplifiers, provides the issuing of m elements of the array $\{b_i\}_{i=1}^{m2^{p-1}}$ .Formation of signals for parallel reading of m elements of the array $\{b_h\}_{h=1}^{m}$ and control instructions for all Sw is carried out by MMU.

SPM works as follows. In each k-th cycle of SPM work recording of m data elements into MU1 and reading of 2m data elements are provided , where first m elements are from the array $\{a_i\}_{i=1}^{m2^p}$, and second m-elements are from the array $\{b_h\}_{h=1}^{m}$. After this, on the basis of the information from control input, calculation for (k+1)-th of parallel reading of m elements from array $\{a_i\}_{i=1}^{m2^p}$ is performed in MMU. Address of MU with the minimum element for (k+1)-th reading is calculated using the formula:

$$A_{(k+1)\,0} = A_{k\,0} + z_k,$$

where ?? - address of the minimum element in the k-th reading; zk – shift of the elements of array ??, that comes in MMU from the input of control signals C. In (k+1)-th cycle of work on (m+1),..., 2m outputs of SPM data will come from MC, with numbers $A_{(k+1)\,0}$,..., ($A_{(k+1)\,0} + m$-1). Also two groups of control signals for managing of the switches of KE are formed in MMU. At that, switches with numbers 1, ..., $A_{(k+1)\,0} + m$ are set to transfer data elements with a shift by m elements and switches with numbers ($A_{(k+1)\,0} + m + 1$), ..., Q are set to transfer data with a shift by z elements. With the arrival of a tact impulse on the TI input writing of information from inputs of according switches and reading of data to the outputs 1,...2m is performed in all MC. It is necessary to note that on the 1-th,...m-th outputs of SPM m ordered data elements from array $\{a_i\}_{i=1}^{m2^{p-1}}$ are read, the biggest of which goes to the first output, next – to the second output and so on, and on the (m+1)-th, … 2m-th outputs – m data elements from the array $\{b_i\}_{i=1}^{m2^{j-1}}$, which is arranged in descending order with cyclic shift to the left on $L = (A_{(k+1)_0} + m - 1)_{\mathrm{mod}\, m}$ words.

For each p-th PU the amount of SPM is determined by the formula $Q=(2^{p-1}+1)m$.

V. CONCLUSIONS

1. The development of highly efficient parallel structures for real-time merge sorting of intensive data streams would be best to carry out at an integrated approach, witch covers methods, algorithms, structures and VLSI technology and take into account the peculiarities of particular application.

2. In the matrix device for data sorting to achieve consistency of data input flow intensity with sorting intensity you can change digit capacity of data input channels, digit capacity of PU and use of hybrid algorithms, based on the combining of merge sort and counting sort.

3. In the flow data sorting device consistency of data input flow intensity with sorting intensity is achieved by changing of numbers of two-way merging channels.

4. New algorithms and structures of devices for parallel and parallel-flow merge sorting of intense data flows in real-time have been developed, in which by means of hybrid algorithms, changing of the number of channels and digit capacities of input data flow consistency of input data flow intensity with sorting intensity is achieved, providing increasing of the efficiency of equipment use.

REFERENCES

1. D. Knuth. Art of Programming, Volume 3: Sorting and searching, 2 pb. -. M, 2000.-832 with.

2. High computing for multiprocessors and multicore Systems // Publisher Moscow University, 2010. - 544 p.

3. Tsmots I.G., Antoniv V.Ya., Parubchak V.O. Parallel - vertical sorting by merge sort with counting sort. Collection of scientific works. The Institute of modeling issues in Energy. Issue 68, 2013. p. 92 - 100.

4. Tsmots I. G., Rahman M. L. Algorithms and devices of parallel – streaming sorting. Collection scientific papers Institute of modeling issues in Energy. Journal of Kyiv 2001. 21. C. 183-191.

5. Parallel computing on the GPU, architecture and software model of CUDA // Publisher Moscow University, 2012. - 336 p.

6. Tsmots I.G., Demido B. A. Structures of specialized parallel memory with highly processor management and digital processing signals // Herald RC "Lviv Polytechnic "," Computer engineering and information technology ", № 380. - Lviv, 1999. - c.18-29.

7. Grushitsky P. AND., Mursal A. H., Ugriumov E. P. Design systems on chips of

ISSN: 2278 – 1323

programmable logic. - SPb: BHV - St. Petersburg, 2002. - 608 p.

8. S.Kun. Matrix processors on VLSI: - M:. Mir, 1991.-672

9. A... S. 1298737 (USSR). The device for sorting numbers. A. A. Miller, I. D. Tsmots / Bul. Inventions 1987, № 11.

10. Computers on VLSI: At 2 Books. Book. 2. Motooka T., Horikoshi X. And . al-M: Mir, 1988 - 336 s.

11. Tsmots I.G. Information technology and specialized tools for processing signals and pictures in real time. Lviv: UAH works, 2005.- 227c.

12. Tsmots I. G., Rahman M. L. Parallel algorithms and devices sorting numbers // Collection of science papers IPM NAS Ukraine, Vol. 11. - Kyiv, 2001. - C .83-91.

13.Ukraine patent on invention №29700. The device for sorting numbers. Batiuk A.E., Rashkevych Yu. M., Tsmots I. G. 2000, Bull. № 6-11

14. Ivan Tsmots, Bohdan Demidov. Memory structures with discipline access FIFO // "Lviv Polytechnic "," Computer engineering and information technology " Number 386. - Lviv, 1999. - c .21-26.

Ivan Tsmots – professor and doctor of technical science in Lviv Polytechnic National University.
Oleksa Skorokhoda – PhD in Lviv Polytechnic National University.
Volodymyr Antoniv – postgraduate in Lviv Polytechnic National University.

ISSN: 2278 – 1323