3 Design and Application of the PLD-Based Reconfigurable Devices

Alexander V. Palagin and Vladimir M. Opanasenko

Department of Microprocessor Devices, Institute of Cybernetics,
Ukraine, Kiev
e-mail: {Palagin_a,vlopanas}@ukr.net

Abstract. Theoretical bases of construction and designing of the PLD–based reconfigurable devices, including the new formalized design techniques of construction and dynamic reconfiguration of architecture and structure of digital devices with a high degree of reconfiguration, corresponding with properties of performing algorithms, constructive and technological features PLD, and also tool means of their designing, are presented. Bases of the theory of adaptive logic networks, intended for the solution of a wide class of tasks by direct structural realization of algorithms of processing and direct representation of input data to output data by functional and structural customization for universal components of a network, are developed. Synthesis algorithms of adaptive logic networks on the classes of tasks set are developed. Design techniques of the computer systems with using of the standard CAD PLD (ISE Foundation) are developed. The structure of the reconfigurable computer system with the open library of configuration files for basic parametrical blocks, including the threshold device, Hemming adder, sorting devices, median filters, matrix multipliers etc. are designed.

3.1 Introduction

The level of development and manufacture of products of high technologies among which one of leading places is occupied by tools of computer engineering (CE), appreciably defines technological progress of many industries. Now scientific researches and practical development in the field of CE on perspective element base, i.e. microprocessors, microprocessor complete sets and systems on the chip in a combination to the LSI circuit of memory and Programmable Logic Devices (PLD) are called to satisfy requirements of the broad audience of users are carried out and to put in pawn bases of development of new effective means of computer engineering.

Development of batch production VLSI demands greater expenses both for development, and for the equipment for their manufacturing. In the schemes realized by a method of printed circuit, change to bring difficultly enough, and in the schemes executed in the form of the LSI and VLSI, any, in advance not stipulated changes, are impossible in general. It not only limits opportunities of their specialization for concrete applications, but also prospects of modernization, expansion with addition of new functions, modification in algorithm of functioning. Therefore one of actual requirements to modern digital systems and devices is increase of their adaptability (flexibility). The basic direction of increase of an adaptability of devices (systems) now is specialization of digital devices by programming their structure.

The first theoretical researches, devoted to synthesis reconfigurable devices, concern to the beginning of 60th years. Base work [1] on the organization reconfigurable computer, presented it as two basic parts: a constant – a computer with fixed structure, and a variable – in the form of a set of computers which can reconstruct the structure by means of the program. It promoted occurrence of a new direction in computer facilities on designing reconfigurable devices with virtual (programmed) architecture on the basis of PLD – Reconfigurable Computing (RC). The term “Reconfigurable Computing” generally designates two-uniform concept: as reconfigurable structure of a computer (hardware), and the process of data processing which is performed by a computer. The significant contribution to development of the given problematic was brought also with works [2–6].

Also subject domains in which reconfigurable computer systems (RCS) have found the "lawful" niches were defined and continue to develop intensively. It is first of all:

- The hardware systems guaranteeing safety of control by especially important objects;
- Complex physical experiments with modeling and management in real time;
- Effective digital processing of high-frequency signals;
- Acceleration of tool means of the automated designing of objects of new techniques and technologies;
- Emulation and designing of wireless communication systems, etc.

Importance and perspectives of the specified scopes testify to urgency of direction Reconfigurable Computing and the problems connected with development of technology RC.

Application of PLD gives an opportunity to realize structures of devices with dynamic reconfiguration and by that to solve problems of effective adjustment for the set algorithm, survivability and reliability. *Reconfigurability* – property of system to redefine set of hardware and connections between them in conformity algorithm of functioning. New physical principles and technical opportunities of microelectronic components are, in turn, a source of new principles of construction and new architecture of modern means CE.

Most a wide circulation has received PLD two types: *CPLD* – Complex Programmable Logic Devices; *FPGA* – Field Programmable Gate Array [7].

CPLD consists of set of PAL–like functional blocks (36V18) which contain macrocells and incorporate a matrix of switching to blocks of input-output. Use of FastFLASH-technology allows realizing intrasystem programming with non-volatile storage of configuration data. Feature CPLD is predictability of delays of the signals.

Architecture FPGA generally represents a matrix of logic cells – configurable logic blocks (CLB), surrounded by peripheral cells – Input/Output Blocks (IOB). Connections between cells are carried out by means of programmed matrixes of interconnections. Everyone CLB contains the combinational logic part, remembering element and internal blocks of management and trace. A basis of combinational part CLB is high-speed static CMOS memory and for realization any Boolean functions the technology of Look-Up Table (LUT) is used and the delay of distribution of a signal through the combinational block is independent of generated function. Programmed interconnections provide all communications inside of a crystal. IOB provide the interface between contacts of a crystal and its internal components.

Under existing forecasts, crystals Virtex series by present time should reach logic capacity up to 100 million logic gates (at initial 50 thousand). If first crystals FPGA were manufactured on technology 0,34 microns, now – 65 nanometers.

FPGA series of type Spartan and Virtex [8] possess similar architecture. PLD the considered series except for the logic sold in logic cells (LC), contain block memory (BR) which unlike the distributed memory sold on logic cells, is built in and does not borrow logic resources. Memory BR is organized in the form of blocks, each of which represents the two-port synchronous device of various capacities depending on type FPGA. To modules of a general purpose, except for BR, multiplier units, and built in receiver-transmitter blocks are entered into series Virtex-II Pro with speed of the transfer reaching some Gbit/sec on the channel in a duplex mode, and also RISC-processors blocks of PowerPC type.

3.2 Evolution of Computer Systems

The RC grows out evolution of computer systems (CS). One of the important stages of evolution is creation of emulating computer systems, possessing ability of modification and full change of internal language. The concept of flexibility of architecture CS has been formulated.

In particular, has been developed and has received practical approbation logic–information method (LIM) designing of the microprocessor systems, uniting in itself theoretical concepts of the theory of digital automatic devices and theories of the information. Essence of LIM is illustrated by the scheme:

$$\begin{tikzcd}
\forall_i(\exists\Omega : A_N \Rightarrow & \Lambda_N \arrow[d, dotted] & \Rightarrow R_N \\
A_i \Rightarrow & \Lambda_i \arrow[d, dotted] & \Rightarrow R_i \\
A_0 \Rightarrow & \Lambda_0 & \Rightarrow R_0(\Theta = \Theta_{extr}(Q,t))
\end{tikzcd}$$ (3.1)

.

where: $A_i, \mathit{\Lambda}_i, R_i\ (i = \overline{N,0})$ – accordingly sets of algorithms, operators and their information-code representations at i -th level of programming, Θ – a set of the generalized characteristics (hardware resources (Q), time (t), etc.).

Modern PLD have defined *the new stage* of evolution connected with creation high–efficiency CS. For the formalized representation of model reconfigurable devices updating of method LIM which is illustrated by the following scheme is offered:

$$\begin{tikzcd}
\forall_i(\exists\Omega : A_N \arrow[dd, dotted] \arrow[dddd, dotted, bend right] & \Rightarrow \Lambda_N \arrow[ddl, dotted] & \Rightarrow R_N \\
\vdots & \vdots & \\
A_i \arrow[dd, dotted] & \Rightarrow \Lambda_i \arrow[ddl, dotted] & \Rightarrow R_i \\
\vdots & \vdots & \\
A_0 & \Rightarrow \Lambda_0 & \Rightarrow R_0 \ (\Theta = \Theta_{extr}(Q,t))\ .
\end{tikzcd} \tag{3.2}$$

In the scheme (3.2) for classical architecture following levels of programming are used: τ_0 – physical or "zero"; τ_1 – microprogrammed; τ_2 – programmed; τ_3 – algorithmic. Programming at a "zero" level defines physical structure of the device which finally realizes the set algorithm of functioning, i.e. carries out programming structure of the device. In difference from the scheme (3.1), updating (3.2) realizes not microprogrammed, but hardware realization of algorithms on gate level. In it difference reconfigurable devices with programmable structure from modern computers consists. Thus the logic structure reconfigurable devices can dynamically vary both by preparation for the decision of a problem, and during computing process.

The most widespread requirement shown to facility CS, high speed is. The given problem, in particular, arises at use of means CS for problems of management and modeling. At use of computers for control of moving objects, technological processes, fighting operations, etc., they should work with anticipation of real processes in operated object or, and generally speaking, in real time. There is a class of problems in which it is necessary to operate quickly the changing processes proceeding in short time intervals, and highly dynamical, quickly functioning objects. Thus simultaneously with high speed maintenance of high accuracy of management is required. Therefore the computer for maintenance of high speed and accuracy of management should possess ultrahigh speed to provide simultaneously set accuracy and work in real time. The similar problem arises and at use of facility CS for modeling complex dynamic objects, and also quickly proceeding processes and the phenomena.

With the advent of modern crystals FPGA began possible to use the results received earlier for construction reconfigurable devices and systems of the raised complexity on the basis of PLD with completely programmed architecture. One of approaches for increase of productivity of facility CS is the combination conveyorization and parallelism.

If frequent change of carried out functions down to new function takes place at each new execution, the conveyor with a dynamic configuration takes place. The given approach is realized in technology RC. As an example of such realization hypercomputer HAL (Star Bridge System Corp.) which is constructed on crystals FPGA can serve. During functioning crystals change the structure and functions is continuous at the decision of numerous computing problems in a mode of real time.

Opportunities RC are introduced in supercomputers Cray XD1 with the purpose of increase of productivity for target appendices by use of a subsystem of acceleration of the appendices, based on crystals of type FPGA (Virtex–4) firm Xilinx which can be programmed on acceleration of key algorithms, such as search, sorting, digital processing of signals, etc. the Given subsystem functions as the coprocessor in relation to base processor AMD Opteron.

In Berkeley Wireless Research Center supercomputer system High-End Reconfigurable Computing System (HERC) on crystals FPGA [9] is developed. Basis of HERC is system prototype BEE (Berkeley Emulation Engine) with two modules, intended for designing, construction and programming HERC for of some applied areas. In opinion of developers, use BEE the system based on processors DSP with similar power consumption and cost, and more than on two orders in comparison with the systems realized on the basis of standard microprocessors provides on the order greater productivity, than. The main components of architecture – computing blocks and the programmed communication environment. The computing block is structurally presented in the form of the printed-circuit-board, which contains four crystals FPGA – processing modules with memory (up to 4 GB everyone) and one for management (the operating module).

3.3 Architecture and Structure of PLD-Based Computer Systems

The typical reconfigurable *computer system* (CS) consists, as a rule, of 2 parts: constant (or "fixed") part F – a Host-computer and a variable part V – reconfigurable subsystem (RSS) which can be united in various configurations. The architecture of reconfigurable systems depends on capacities of sets of algorithms: (N_F), carried out on the equipment F, and (N_V), carried out on the equipment V. The parity of these sizes defines offered classification of reconfigurable computing systems:

a) The computing systems focused on a Host-computer in which the basic computing capacities are concentrated, and reconfigurable computer provides increase of productivity only for a narrow class of problems ($N_F \to N, N_V \to 0, N_F >> N_V$);

b) The computing systems focused on RSS in which the Host-computer is used, basically, for performance of auxiliary functions (service, input-output), and all algorithms are carried out mainly in RSS which can have own field of external devices (through payments of expansion) or the general field of external devices with a Host-computer to which RSS has direct access;
c) Reconfigurable computing systems in which a Host-computer and RSS have approximately identical complexity, thus RSS it is focused on the decision of labour-consuming problems, and the Host-computer provides strong support regarding translation, input-output, service, etc.;
d) RSS is the independent device in case of $N_F = 0, N_V = N$, and the Host-computer is absent.

RSS connects to a Host-computer through one of the standard trunks, the variants of connection most widespread today are realized through trunks PCI and PCI–Express. RSS have functional processing field (FPF) the set dimension which is configured for performance of the set algorithm or its part, providing, thus, optimum realization of this algorithm both under time characteristics, and on hardware expenses.

Introduction in practice of crystals PLD and HDL–technology (Hardware Description Language) for performance of projects in this element basis intensified development of a wide spectrum of the digital modules representing ready technical decisions, essentially reducing time of designing and an output for the market of new products. Such opportunities of HDL–technology as hierarchical designing, bearableness of libraries, platform- independence, allow using available soft cores as macrocells for development of new technical decisions. The architecture of modern crystals FPGA is optimized for use both hard and soft cores, and allows integrating them into projects easily. For example, crystals of Virtex type have in advance built in multipliers and PowerPC processors as hard cores, and also other functional blocks.

In RSS, or devices with programmable architecture the functional field of the set dimension configured specially for performance of certain set algorithm or its part is fixed, providing, thus, realization of this algorithm optimum, by the set criteria, by way. Adjustment of structure for performance of demanded algorithm and its realization in a crystal on a gated level allow increasing speed of the device by some orders in comparison with universal decisions.

The algorithm can be broken into the fragments which are carried out consistently in this connection, structures corresponding these fragments also are loaded into a crystal consistently (by way of their performance), that leads to essential economy of resources. Complexity of fragments of algorithm thus is defined by only logic capacity of a crystal, i.e. dimension of a processing field.

Thus, reconfigurable data processing represents to a certain extent change of the central paradigm of designing of modern means of computer facilities.

The model of the projected computing system is offered:

$$S = <M, A, B, P>,$$

where: M – set of mathematical methods, characteristic for a subject domain, reflecting functioning of system; A – set of algorithms of realization of a method;

$B = \{b\}$ – the components of alphabet from which the structure is synthesized; P – procedure of the description of the project (the description of object). Thus, process of designing consists in the decision of a problem of synthesis of structure on the basis of components $\{b\}$ the alphabet B for performance of the certain algorithm A realizing a method M, underlying functioning of structure, according to requirements of specifications. Result of procedure P is the description of the project by means language CAD.

Synthesis of structural realization of sequence of algorithms is offered, when the method/problem (M) is represented sequence of algorithms ($A_i, \forall i = \overline{1 \div n}$):

$$M = \bigcup_i A_i \,.$$

In RSS, the base (zero) architecture realized on chip of PLD in the form of a functional processing field of fixed dimension, the controller of the trunk of a host-computer, a field of memory, and also well structured library of configurations files (LCF) structural realizations of the methods (algorithms) which are carrying out display of algorithm in structural realization ($F: A_i \Rightarrow B_i$) is initially set. Each algorithm has display $F: A_i \Rightarrow B_i$ in structural realization (B_i) which represents a configuration file for a crystal PLD. Generally there are some variants of realization of algorithm (for example, consecutive, series-parallel and parallel):

$$B_i = \bigcup_z B_{iz}, (z = 1 \div k)\,.$$

Each variant is characterized by parameters of speed (time of performance – t_{iz}) and hardware expenses (q_{iz}). And we assume, that capacity of set B is sufficient for realization of a wide set of algorithms. In the event that demanded realization of *i*-th algorithm in library is absent ($B_i = \varnothing$), it is necessary to create by means of CAD PLD it and to include as a standard element in library. Thus, the problem of optimization is reduced to the ordered purpose to each *i*-th top the column of sold algorithm (B_{iz})-th element of library with the purpose of reception of extreme value of some criterion of quality. I.e. any operator is displayed only by one element from library. The structure realizing set columns is as a result defined. Then the decision of a problem can be received by methods of integer mathematical programming and, depending on demanded criterion of quality, it is possible to define following variants of statement of a problem of optimization.

The problem of optimization consists in definition of a minimum of criterion function, and criteria of quality are total hardware expenses for realization of all algorithms:

$$\alpha \sum_i \sum_z t_{iz} x_{iz} + \beta \sum_i \sum_z q_{iz} x_{iz} = min, \quad (\forall i = \overline{1 \div n}, \forall z = \overline{1 \div k}),$$

under conditions of restrictions $\sum_{z=1}^{k} x_{iz} = 1$, $\sum_{i}\sum_{z} q_{iz}x_{iz} \le Q_0$ $\sum_{i}\sum_{z} t_{iz}x_{iz} \le T_0$,

where α, β – weight coefficients which can be certain, for example, a method of expert estimations; x_{iz} – *z*-th realization of *i*-th algorithm A_i; T_0 – admissible time of performance of all algorithms; Q_0 – admissible hardware expenses.

Methods of the decision of such problems are well enough developed and allow receiving for admissible time the comprehensible decision.

Presented approaches are put in a basis of the generalized algorithm of designing PLD-based reconfigurable devices which represents system of the interconnected algorithms, the part from which is formalized and shown to statement and the decision of a problem of synthesis and a choice of optimum structural realizations from set, the others use heurism. Each algorithm is a separate fragment of designing to which the certain section of the dissertation where it is presented in the form of the formalized technique of designing with a theoretical substantiation of its basic positions and the description of methods of the decision of concrete applied problems of the analysis, synthesis and optimization of separate structural realizations is devoted.

The algorithm of designing of the structural realizations RSS representing a Basic board (for the coprocessors connected to the standard Bus of a Host-computer) or Carrier board (for independent devices) with a set of expansion boards and expansion modules, or crystal PLD for realization SoC (System–on–Chip) is developed. The algorithm represents sequence of stages (Fig. 3.1).

The analysis of problem area statement of a problem a choice of suitable algorithm (in case of absence of a configuration file for realization of corresponding algorithm its synthesis with the subsequent record in LCF) from LCF imaging at a level of the general architecture (function chart) preparation of the formalized technical project programming of structure on the basis of a configuration file a programming of algorithm the decision of a problem an estimation of characteristics of parameters (structure is carried out, process of the decision) check of parameters on conformity to the established criteria (if necessary following iteration) commissioning. The block diagram of algorithm (Fig. 3.1) provides also correction of criteria. It is analyzed features of designing of digital devices on the basis of PLD with use of HDL-technology and CAD PLD.

The developed technique of designing, leaning on the given system of algorithms and logical-information model RSS laying in its basis, allows to decide – to formalize the main task of designing process of search of optimum pair «algorithm-structural realization». The technique intends for designing: the task-oriented coprocessors and the independent devices working with the algorithms set; reconfigurable processors with conveyor data processing; parametrical IP-Core for realization of the algorithms set which are represented by elements of library of configuration files; SoC. It can be modified depending on the initial task, a class of problems, element -technological base, etc.

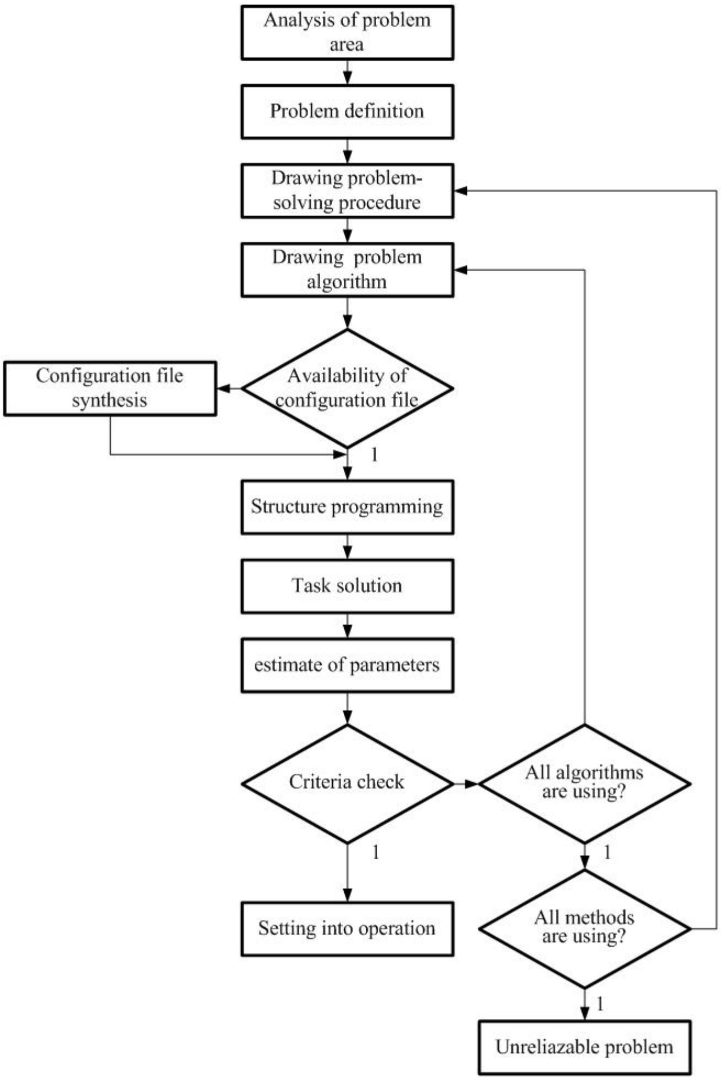

Fig. 3.1 Algorithm of designing of the structural realizations RSS

3.4 Adaptive Logical Network (ALN)

The adaptive logic network is a discrete converter of codes of type of the asynchronous combinational automatic device, set directed graph which tops are logic functions, and edges – communications between them type "output–input".

From the point of view of topology of system ALN represents a matrix of universal logic elements (LE) which are grouped into functional units (FU) and blocks (FB) which site is fixed, thus change of their functioning occurs depending on a class of problems and from their purpose.

Universal LE we shall name the combinational automatic device: $L = \langle n, F \rangle$, where: n – quantity of binary inputs or dimension of entrance variables LE; $F = \{ f_\rho \}, \quad \rho = [1 \div 2^{2^n}]$ – the totality set of Boolean functions. Universality LE consists in an opportunity of its adjustment for realization any Boolean functions.

Structure ALN can be described by following system:

$$A = \langle n, h, F, S, L, m, D, X, Y \rangle,$$

where: n – word length of input binary vectors (dimensionality of ALN on an input); h – target word length ($h = \overline{1 \div n}$), dimensionality of ALN on an output; $F = \{F_{ij}\}$ – set of logic functions of system; S – structure of communications between LEs; $L = \{L_{ij}\}$ – set LE (i –a serial number of element LE; j – number of a level of processing); m – quantity of levels of processing; $D = \{d\}$ – set of n-dimensional binary vectors (training sample); X – full set of input binary vectors; $Y = \{Y_{ij}\}$ – the generalized function of system, $Y_{ij} = f_{ij}(Y_{v,(j-1)}, Y_{w,(j-1)})$ – value of the function f_{ij} sold by an element L_{ij}, $Y \in \{0, 1\}$ which structure is resulted on Fig. 3.2 (v, w – value of an index i for inputs LE).

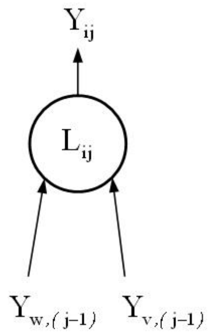

Fig. 3.2 Structure of universal logic element

Each level ALN represents a ruler of LE (ρ inputs for all), each of which can be adjusted on performance of any of a full set (2^{2^ρ}) Boolean functions of its input variables and realizes imaging of l-dimensional $(l \le n)$ binary vectors into a u-dimensional $(l \ge u)$ binary vector. Matrix of LEs or FU represents the combinational automatic device without the memory, a having l-digit input, a u-digit output and m – quantity of lines of a matrix. Within the limits of one level the type of function can be set for everyone LE separately (stepwise adjustment) or for all LEs (by the level adjustment).

The functional block represents a network of consistently included automatic devices (hierarchical assembly of l ($l = \overline{1 \div m}$) functional units). In the further we shall be limited to consideration of three types FB distinguished to a topological attribute: «rectangular matrix» (RM) – ($h = n$); «triangular matrix» (TM) – ($h = 1$); «trapeziform matrix» (TrM) – ($h = \overline{2 \div (n-1)}$). Depending on structure of communications following TM types are offered: with logarithmic structure of communications (LSC); with cellular structure of communications (CSC); with asymmetric structure of communications (ASC). The offered structures of communication differ on capacity sold Boolean functions and to hardware resources.

Problems of the structural organization and synthesis of multilevel structure ALN of type TM consists in definition of types of logic functions f_{ij} for all LEs network. For definition of set of logic functions $F = \{f_{ij}\}$ the approach based on the description of a Boolean network by polynomials which factors are set, in particular, by means of Adamaar matrixes is used [10].

At coding values Boolean functions and its arguments transition to coding with use of values (1) and (–1) is carried out. Thus, the set of variables $X = \{x_1, x_2, ..., x_n\}$ for Boolean from n variables will be represented function f by set $E = \{e_1, e_2, ..., e_n\}$, where $e_i = (-1)^{x_i}$, and set of values $Y = \{y_1, y_2, ..., y_{2^n - 1}\}$, where $y_j \in \{0, 1\}$; set $V = \{v_0, v_1, ..., v_{2^n - 1}\}$, where $v_j = (-1)^{y_j}$. For any Boolean functions f from n the variables accepting values from set $\{1, -1\}$, there is an equivalent polynomial $P_{f(n)}$ with factors from set of real numbers– $f(X) = P_{f(n)}(X)$. Factors of a polynomial for function f enter the name by means of Adamaar matrix ($A = \frac{1}{2^n} H_n V_n$,), where $A = \{a_0, a_1, ..., a_{2^n - 1}\}$ – set of factors of a polynomial, H_n – Adamaar matrix dimension 2^n, V_n – set of values Boolean functions.

By way of illustration applications ALN of type TM a number of functional devices of the average complexity focused, mainly on problems of recognition of images is synthesized.

So, on the basis of the scheme of transformations using as base bit operations of addition and multiplication on the module 2 (logic operations XOR and AND) the problem of synthesis of adder Hemming of any word length is solved [11]. The offered synthesis algorithm of adder Hemming of any word length carries out imaging: $\Im : (g, d) \Rightarrow T$, where: $g \in G, d \in D, T = \sum_{l=1}^{n} (g_l \oplus d_l) = \sum_{\lambda} 2^{(\lambda - 1)} \tau_\lambda$; τ_λ – a component of the vector, containing value λ–th bit of binary representation T of a mismatch of vectors g and d; $\lambda = 1 \div (\mathrm{Ent}\{log_2(n+1)\})$; $2^{(\lambda-1)}$ – weight λ-th bit of binary representation of a mismatch T.

The problem of synthesis of the threshold device of the any word length realizing threshold operation from set $\Psi = \{\psi_\xi\}$ is solved also, where $\xi = 1 \div 4$ (ψ_1 –operation $\le$, ψ_2 – operation $\ge$, ψ_3 – operation>, ψ_4 – operation <), types of logic functions for each level of structure TM (type ASC) depending on value of a threshold are as a result defined.

The synthesis algorithm of the threshold device is based on the bit-by-bit analysis of value of a threshold vector Θ according to binary data presentation: $\Theta = \sum_\lambda 2^{\lambda-1}\theta_\lambda$, where $\tau_\lambda, \theta_\lambda$ – the components of vectors containing value λ-th bit of binary representation T and Θ accordingly, and $2^{\lambda-1}$ – weight of λ-th bit ($\lambda = 1 \div n$, n – dimension (length) of a binary vector).

The synthesis algorithm of the symmetric threshold device of the any word length realizing symmetric threshold operation with the top and bottom borders, symmetric concerning the center of a numerical axis on a piece $[0 \div (2^n - 1)]$ is developed also. As the set operation concerns to symmetric functions at the first level of structure TM logic function XOR is used, and for other levels types of logic functions F_s^ξ are defined to algorithm similarly considered above for the threshold device.

3.5 Problem-Oriented Structures of Digital Devices

The technique is developed and process of designing the typical reconfigurable problem-focused devices with hardware PLD-based realization in the form of base library parametrical functional blocks by means of their VHDL language description and the Schematic editor is considered. Library of functional devices of the wide application providing use by the broad audience of developers at designing of digital devices by the task of corresponding parameters and a choice of optimum structure (by criteria speed-complexity of realization) are developed.

The following developed functional blocks are included in structure of library of files of configurations: Hemming adders which are carrying out calculation of distance Hemming for 4, of 8 and 16-digit numbers; two variants of realization of algorithm of sorting (the linear sorter and the memory-based sorter); multiplier square matrixes of the order $m = 10$ for the whole 16-digit numbers; the median filters using consecutive, it is serial–parallel and parallel computing models; arithmetic devices of multiplication with a floating point of unary accuracy (compatible to standard IEEE–754). The developed functional blocks are verified at real stands and the reconfigurable device (board ADS–XLX–SP3–EVL400) that proves their functioning.

3.5.1 Functional Blocks with a Floating Point

During designing of mathematical coprocessors, the DSP processors, the in-built arithmetic coprocessors wide application is found with floating point functional blocks. Many vendors (for example, Nallatech Corp.) are developed own soft cores for realization of such arithmetic operations, has developed Core for processing operands with a floating point (standard IEEE-754) under Virtex series.

The problem of designing of arithmetic devices and algorithms for processing operands in a floating point format is actual and now. Standard IEEE–754 gives most the general representation for numbers with a floating point in modern computers, including Intel PC, Macintosh and majority Unix platforms.

Let's consider development of the devices which are carrying out the floating point operations in conformity with standard IEEE-754. The generalized structure of functional blocks with a floating point (Fig. 3.3) and contains of three compound modules: the module input arguments checking module (IAC); the functional module (FM) and the result creation module (RCM). The description of modules is executed by means VHDL language, by development synthesizer FPGA Compiler II from Synopsys is used, system CORE Generator System is applied to formation of IP-Core blocks. The developed modules are verified by a modeling method with definition of time and hardware parameters. Modules represent the finished typical technical decisions and can be used in other projects as soft cores.

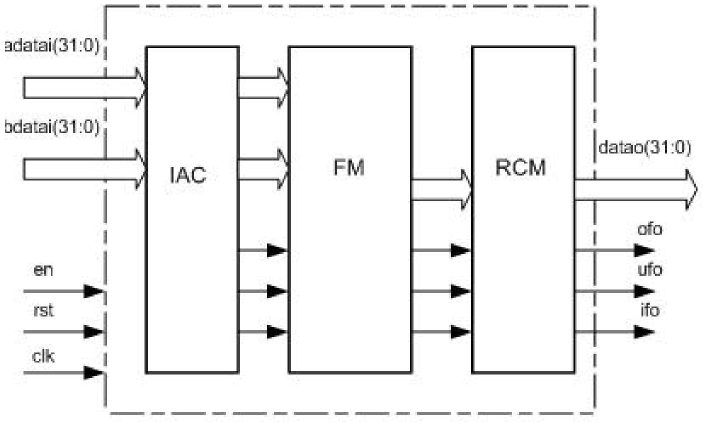

Fig. 3.3 Structure of the functional block with a floating point

IAC will transform input data, analyzes them on conformity to standard IEEE-754 with formation of corresponding attributes. Corresponding numbers and the information concerning classes of input data gives out as results to the functional module with a floating point.

FM carries out the set operation from a floating point with formation of corresponding attributes.

RCM carries out the conformance of a format result data with standard IEEE-754 and final setting of flags.

Inputs and outputs of the floating point block are not adhered to the fixed input-output contacts of a specific FPGA, because using of any chips therefore is supposed. Assignment of inputs and outputs of the block is shown on Fig. 3.3: clk – a global signal clock; rst – a global signal reset; en – Enable signal; adatai (31:0) – the input data bus A; bdatai (31:0) – the input data bus B; datao (31:0) – the output data bus; ofo – a flag “Overflow”; ufo – a flag “Underflow”; ifo – a flag «Inadmissible operation».

For the agreement of the obtained result of transformation with standard IEEE–754 it is necessary to present numbers as normalized form. Therefore it is required to define a high-order digit «1» and to realize shift aside to the high-order digit on demanded number of bits with simultaneous subtraction of this value from the resulting exponent part. Presence of powerful logic resources in crystals Virtex series allows accelerating this procedure by fast definition of number of shifts. Then, unlike realization of serial shift with the simultaneous analysis of the high-order bit, is carried out parallel shift on demanded number of position for normalization of a mantissa.

Floating point Addition a includes strictly serial five operations: comparison of exponents, shift to the right mantissas of smaller number, summation of mantissas, search of left unit of a mantissa of result, normalization of a resulting mantissa.

For realization of operation of search of left unit using priority coder is offered. Let is available ($n = 24$) meaning bits of a mantissa. It is required to define number of the high-order "nonzero" position and to carry out of normalization of a mantissa $\mathrm{F} = \{ \mathrm{f}_{23}, \mathrm{f}_{22}, \ldots, \mathrm{f}_{\mathrm{i}}, \ldots, \mathrm{f}_0 \}$.

Priority coder represents the combinational scheme, having n inputs and ($\mathrm{Ent}\{log_2 \mathrm{n}\}$) outputs which consists of two sequentially connected schemes – the first allocates high-order unit, and the second its number (number of demanded shifts) in an operand.

The first scheme has n inputs and n outputs, realizing following system of the logic equations:

$$\mathrm{a_i} = \mathrm{f_i}\left(\bigcap_{\mathrm{i=i+1}}^{(\mathrm{n-1})} \overline{(\,\mathrm{f_i}\,)} \right) \forall \mathrm{i} = 0 \div (\mathrm{n}-1) \qquad (3.3)$$

The second scheme has n inputs and ($Ent\{\log_2 n\}$) outputs, realizing following system of the logic equations:

$$\mathrm{y_j} = \bigcup_{\mathrm{k=1}}^{\mathrm{N=Ent}\{(\mathrm{n-1})/(2^{(\mathrm{j+1})})\}} \left[\bigcup_{\mathrm{i}=2^{\mathrm{j}}+(\mathrm{k-1})\times 2^{(\mathrm{j+1})}}^{2^{\mathrm{j}}+(2^{\mathrm{j}}-1)+(\mathrm{k-1})\times 2^{(\mathrm{j+1})}} \mathrm{a_i} \right], \qquad (3.4)$$

$$(\forall \mathrm{j} = 0 \div (\mathrm{Ent}\{log_2 \mathrm{n}\} - 1))/$$

Thus, mathematical expressions (3.3) and (3.4) allow to synthesize priority coder for any word length, the representing parametrical module which can be used by development of new projects by other users.

By development of typical modules as well as by development of usual projects, use already well fulfilled accessible IP–Core is expedient.

Let's consider an example of designing of the 32–bit floating point block of multiplication. The block consists of three elements, first two of which, according to Fig. 3.3 enter into functional module FM, and the third – in functional module MFR.

The first element forms 24-digit operands for the block of multiplication (“1” in the high-order – 23-rd position and 23 digits of fraction of a mantissa), summarize exponents of multiplied numbers (8 bits) and defines a sign on result.

The second element carries out operation of multiplication and is formed by means of the Core Generator (Xilinx Corp.).

The third element carries out check of conditions and formation of result. Following conditions are checked: if the sum of exponents of numbers is equal or more than 255, the signal “Overflow” is formed; if 24-th bit of product of numbers is equal “1”, then shift of product of numbers on one position aside low-order digits and increase in the exponent per unit is made; if, after increase in the exponent per unit, value the exponent becomes equal 255, then the signal “Overflow” is formed.

Using of an element of multiplication of combinational type the result of multiplication is formed on a step following a step of registration of operands. When it is necessary to multiply arrays of the numbers acting synchronously with any clock sequence CLK, using of an conveyor-based element of multiplication is preferable. In the developed module elements of multiplication, both with the multiplier of combinational type, and with 4–levels (LUT–based realization are used) or the 2–level (in–built blocks of multiplication 18x18 are used) conveyor that allows to reduce essentially due to increase in clock speed time of multiplication of arrays of numbers. For the timing agreement four or two series registers in this case are entered into the first element of the module for conveyor transfer on an output of the exponent and a sign of product of numbers. The delay (Latency) between registration of the first operands and registration of the first product of the module of multiplication is equal to 5-th or 3-th periods CLK accordingly at use a 4-level or 2-level conveyor-based element of multiplication.

In Fig. 3.4 the diagram of work of the module of control IAC, executed by means of editor State Editor is represented. On the first step at presence of signal EN=1 the block passes in status STATE1, on which (digits 0–31) from input operand A formed signals EXP_F (exponent – digits 23–30) and FRAC (fraction of a mantissa – digits 0–22).

Further check of conditions is made, at performance of one of which block passes on the second step in one of statuses (STATE2 – STATE6) with formation of a corresponding flag:

- If value of the exponent to equally zero, and fraction of a mantissa nonzero the input operand is nonnormalized number;
- If values of the exponent and to fraction of a mantissa are equal to zero the input operand is zero;
- If value of the exponent is more than zero and less than 255 the input operand is the normalized number;
- If value of the exponent equally 255 and fraction of a mantissa zero, the input operand is infinity ($\pm\infty$);
- If values of the exponent equally 255 and fraction of a mantissa nonzero the input operand is not real number *(NAN).*

Transition of the block in an initial status is made on a signal of reset (RESET) or setting of signal EN in a zero status.

In a chip of series Spartan–II (XC2S50–5) the block borrows 46 Slices and operates on clock speed 103MHz.

Advantage of the offered realizations in comparison with known is reached due to optimum distribution of descriptions of constituent modules in different modes, and also original priority coder which allows to define number of high-order "1" for the subsequent performance of operation of normalization of a mantissa for one timing step.

Comparative estimations of hardware resources and are presented to productivity of the developed modules of the multiplication realized with using Core (Xilinx Corp.), with similar modules of Digital Core Design.

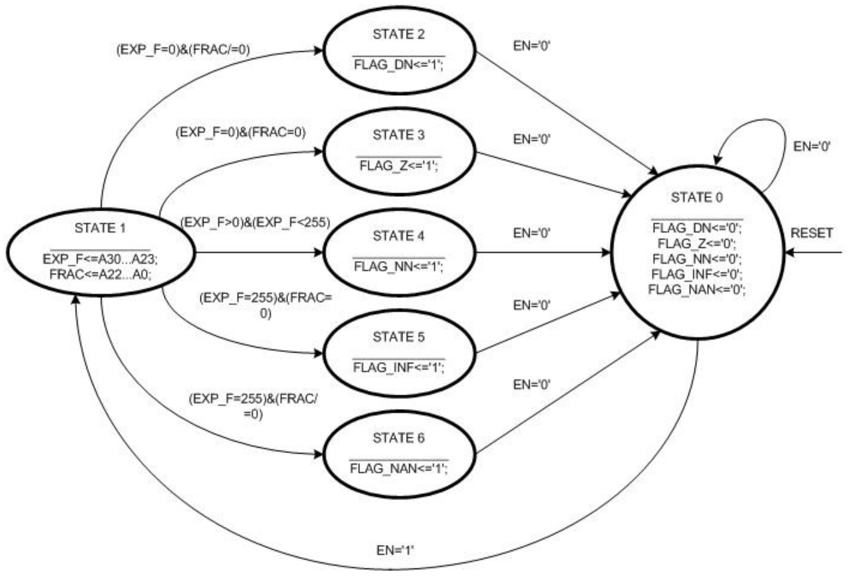

Fig. 3.4 The diagram of work of module IAC

Resources are estimated by quantity Slices. Productivity is estimated by frequency CLK. Hardware resources we shall estimate concerning known realizations:

$$\Delta Q_i = Q_0 / Q_\mu ,$$

where: Q_0 – hardware resources of the module of Digital Core Design; Q_μ – hardware resources of the offered module; Q_1 – hardware resources of the module without the conveyor; Q_2 – hardware expenses of the conveyor-based module.

For a FPGA of type 2S200–6: $\Delta T_1 = T_0/T_1 = 2{,}54$; $\Delta T_2 = T_0/T_2 = 2{,}33$; $\Delta Q_1 = Q_0/Q_1 = 2{,}5$; $\Delta Q_2 = Q_0/Q_2 = 2{,}3$.

For a FPGA of type V300–6: $\Delta T_1 = T_0/T_1 = 2{,}5$; $\Delta T_2 = T_0/T_2 = 2{,}27$; $\Delta Q_1 = Q_0/Q_1 = 3{,}5$; $\Delta Q_2 = Q_0/Q_2 = 2{,}26$.

For a FPGA of type 2V250–5: $\Delta T_1 = T_0/T_1 = 6{,}15$; $\Delta T_2 = T_0/T_2 = 9{,}67$; $\Delta Q_1 = Q_0/Q_1 = 6{,}15$; $\Delta Q_2 = Q_0/Q_2 = 3{,}98$.

The variant of realization of the module on FPGA 2V250–5 with using LUT is absent in offers of Digital Core Design, however regarding hardware resources it we shall compare with offered realizations on FPGA V300–6, but allows to work (approximately on third) with greater clock time.

The synthesized the functional floating point blocks (compatible to standard IEEE–754), can be used as a library element by development of complex computers.

3.5.2 Functional Blocks for Multiplication of Matrixes

One of the basic features of programmable logic is the opportunity of using a principle of parallel data processing at the solving of the wide problems. The increasing of resources of modern programmable logic allows to raise essentially speed of developed devices and to realize by hardware the algorithms working in real time. Multisequencing of calculations or logic operations it can be carried out both at a level of digits of representation of the information, and at a level of the blocks which are carrying out corresponding algorithms of mathematical model. An example of such successful realization is the principle of Parallel Distributed Arithmetic used in digital signals processing.

Let's consider realization of multiplication algorithm of a matrix $A = \left\| a_{ij} \right\|$ of the size $m \times n$ on a matrix $B = \left\| b_{jk} \right\|$ of the size $n \times r$. The resulting matrix $C = \left\| c_{ik} \right\|$ in the size $m \times r$ is formed as follows:

$$C = AB = \left\| a_{ij} \right\| \times \left\| b_{jk} \right\| = \left\| c_{ik} \right\|,$$

Where

$$c_{ik} = \sum_{i=1}^{n} a_{ij} b_{jk} \qquad (3.5)$$

Thus, according to (3.5), each *j*-th element of *i*-th line of a matrix A is consistently multiplied by corresponding *j*-th element of a column of a matrix B and the received products are added.

For definition of each element of resulting matrixes are used operations of multiplication and summation of partial products. Summation can be carried out by two ways: accumulation of partial products at their serial receipt on an input of the accumulator from an output of the multiplier and parallel summation of partial products. The first way assumes presence of the block which is carrying out multiplication and summation (accumulation) of received partial products. The second way uses a set of multipliers and the multiport adder for reception of an element of resulting matrixes.

These ways are realized by several variants:

- Serial (SL), when the processing field consists of one block consistently calculating the sum of pair products in (3.5);
- Parallel-serial (PS1), when the processing field contains set of which quantity correspond to quantity (*i*) lines of a matrix A, by means of which the sums of pair products for elements c_{ij} are simultaneously calculated, and results in (3.5) further are consistently formed;
- Parallel-serial (PS2), when the processing field contains such quantity of blocks, in which quantity of multipliers correspond to quantity (*i*) lines of a matrix A, in parallel realizing, thus, calculation of one element c_{ij} of a matrix C, and further other elements c_{ij} are consistently calculated.

Let's consider realization of the device which are carrying out multiplication of square matrixes of the order $m = 10$ for the whole 16-bit numbers, realized in a crystal Virtex-E series. The quantity of using Slices includes input, output and intermediate registers for realization of conveyor-based calculations. Execution time of operation of multiplication of two 16-bit numbers with accumulation of the 32-bit sum (summation of result of multiplication with the number which is being the accumulator) for specified type of a crystal is 6,424 nanoseconds. In Table 3.1 hardware and time estimations for the considered variants of realization are resulted.

Table 3.1 Results for different implementations

Variant of realization of algorithm of multiplication of matrixes	Quantity of multipliers / adders	Speed (full time of multiplication of matrixes), nanosecond	Hardware expenses (quantity Slices)
SL	1/1	6424	181
PS1	10/10	642,4	1810
PS2	10/1	890	1665

3.5.3 Designing and Realization of Median Filters

Digital methods of processing of images now play a significant role in scientific researches, the industries, medicine, space researches and information-telecommunication systems. One of methods of digital processing the images applied to elimination of defects of the image, caused by handicapes and noise, is the median filtration. Median filters (MF) differed robustness and are convenient for smoothing the information in cases when noise characteristics are unknown. Stepped changes of a signal pass through the median filter without distortion. This feature is used, for example, in the image filtering where data should be smoothed, but distortion of the form of fronts of a signal is inadmissible.

Let's consider realization of the PLD-based median filter, it is using serial, serial–parallel and parallel computing models.

Generally, the median can be defined as magnitude x_{med}, for which at any values z fairly expression:

$$\sum_{i=1}^{n} \left| x_{med} - x_i \right| \le \sum_{i=1}^{n} \left| z - x_i \right| .$$

Median filtration realizes a choice x_{med} for odd n, thus is unequivocal (for even value n there is an infinite number of possible values x_{med}). So, for the two-dimensional window containing 3x3 of elements of the image (pixels), the median filter with nine vectors describing brightness for halftone image or color for the color image, chooses a vector with average value which then is appropriated to central pixel of windows. Median filtration can be carried out also for a window of any other form, for example, crosswise with number of pixels, equal 5 or 9, etc. Irrespective of the form of a window the filter realizes the same algorithm and is characterized by number (n) and word length (m) of processing pixels.

Thus, a median x_{med} of discrete sequence of binary vectors $x_i(i = 1 \div n)$ for odd n is that its element for which exists $(n-1)/2$ elements, smaller or equal to it on size, and $(n-1)/2$ elements, greater or equal to it on size. Let for $\forall x_i \in X$ input set of binary vectors $X = \{x_i\}\ (i = 1 \div n)$ it is necessary to define a median x_{med}. With the purpose of increase of speed of the scheme it is offered to use algorithm of definition of the median, allowing manipulating not input data that is inherent in some algorithms of sorting, and results of comparison of input codes among themselves. The algorithm of definition of a median in this case represents sorting data with the subsequent choice of the code having number $(n-1)/2$ from sorted sequence which numbering begins with zero.

The square matrix is formed:

$$Y = \left\| y_{ij} \right\| \ (i, j = 1 \div n),$$

where $y_{ij} \in \{0,1\}$ – an element of a matrix which is defined by a rule:

$$\begin{cases} y_{ij} = 1, \text{if } x_i \ge x_j; \\ y_{ij} = 0, \text{if } x_i < x_j. \end{cases} \tag{3.6}$$

Elements y_{ij} of the main diagonal of a matrix $(i = j)$ accept zero value. And if $(\forall i, j)\ y_{ij} = 1$, then $y_{ji} = 0$ and on the contrary. Therefore values of elements y_{ij} with the indexes $i > j$, laying above the main diagonal (the quantity of these elements is defined by size $(n^2 - n)/2)$ are defined. Values of the elements y_{ij} $(i < j)$ laying below the main diagonal are defined as follows:

$$\text{if } (\forall i, j)\, y_{ij} = 1, \text{ then } y_{ji} = 0;$$

$$\text{if } (\forall i, j)\, y_{ij} = 0, \text{ then } y_{ji} = 1.$$

For every line received matrix $Y = \|y_{ij}\|$ the arithmetic sum of values of elements y_{ij} is calculated:

$$s_i = \sum_j y_{ij} \tag{3.7}$$

Depending on numerical value s_i which unequivocally corresponds to input vector x_i, we receive result of sorting of set of vectors X:

$$\begin{cases} if\ s_i = 0, \text{ then } x_i = min\{X\}; \\ if\ s_i = (n-1), \text{ then } x_i = max\{X\}; \\ if\ s_i = (n-1)/2, \text{ then } x_i = \text{med}\{X\}. \end{cases} \tag{3.8}$$

The offered algorithm is realized by various ways, depending on quantity of simultaneously formed elements of a matrix $Y = \|y_{ij}\|$.

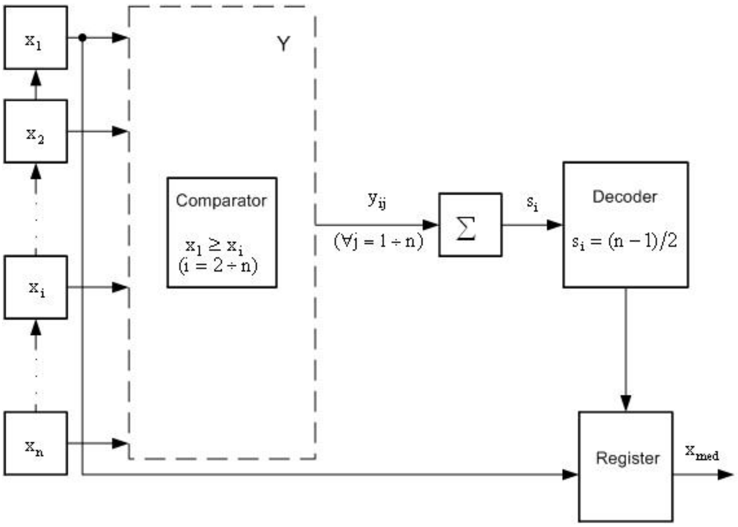

Fig. 3.5 Functional scheme of the median filter SL

At serial realization (SL) it is consecutive on each step of algorithm one comparison is carried out only and one element of a matrix Y is formed. The maximum quantity of steps necessary for realization of algorithm, is equal to quantity of operators of comparison (comparators) plus one step for record of initial data. The quantity of comparators is defined by size $(n^2 - n)/2$. Thus function scheme is presented on Fig. 3.5.

In the given realization it is carried out step-by-step formation of a matrix $Y = \|y_{ij}\|$ by comparison of an input vector $x_i\ (i = 1 \div (n-1))$ with vectors $x_{i+1}, x_{i+2}, ..., x_n$ according to (3.7) and (3.8). The received elements $y_{ij}\ (\forall j = 1 \div n)$ line-by-line are summarized in conformity with (3.9) and the result is compared to a constant equal $(n-1)/2$. In case of equality $s_i = (n-1)/2$ the corresponding vector x_i gets out as a median x_{med} and performance of algorithm stops.

In serial-parallel realizations (SP) for one step it is carried out in parallel from one before $(n-1)$ comparisons, and lines of a matrix Y are consistently formed. The maximum quantity of steps necessary in this case for realization of algorithm, in view of a step of record of initial data equally $(n+1)$. In this realization consecutive formation of a matrix $Y = \|y_{ij}\|$ by parallel comparison of input vector $x_i\ (i = 1 \div (n-1))$ with vectors $x_{i+1}, x_{i+2}, ..., x_n$ according to (3.7) and (3.8) is carried out. The number of comparisons on everyone $(i+1)$–th step of formation of a matrix Y in relation to i–th step decreases on unit. On penultimate $(n-1)$–th step is carried out only one comparison (x_{n-1}, x_n). Elements $y_{ij}\ (\forall j = 1 \div n)$ line-by-line are summarized according to (3.9) and the result is compared to a constant equal $(n-1)/2$. In case of equality $s_i = (n-1)/2$ the corresponding vector x_i gets out as a median x_{med}.

Further we shall consider a variant of construction MF, in which the median is defined for one step.

In parallel realization (PR) all comparisons are carried out simultaneously and elements of a matrix Y are formed in parallel. In the given variant the square matrix $Y = \|y_{ij}\|$ is formed of set of input vectors $x_i\ (\forall i = 1 \div n)$ in conformity with (3.7) and (3.8). Elements y_{ij} are simultaneously summarized line-by-line, according to (3.9), and the received values in parallel are compared to a constant $(n-1)/2$. Equality $s_i = (n-1)/2$ unequivocally defines a choice x_i as a median.

PLD-based realization of the median filter.

Synthesis of structure MF can be executed by means of Schematic Editor and libraries of components Project Libraries of system of designing Foundation Series. Except for opportunities of the description of the scheme in the specified way the system of designing puts at disposal of the designer more progressive means of the description of project HDL Editor and State Editor. First of editors serves for the description of the equipment on one of languages (VHDL or Verilog), the second – for the description of work of automatic devices by means of diagrams of statuses which further are automatically translate into the HDL description.

Essential advantage of the HDL description of the project is the opportunity of the description, both architecture of the projected device, and its behaviour. Besides such description in comparison with using of the Schematic editor is easily modified. Presence of modern the synthesis programs which is carrying out transformation of construction of a HDL-code in the scheme of logic elements, allows to carry out complex projects to similarly development of programs in language of a high level.

Example of serial-parallel realization of algorithm of median definition.
The description of the project is spent with using of construction of VHDL-language for the most simple in the description of variant MF for five 8–bit pixels.

Let's present MF as a "black" box on which input acts five 8–bit vectors: *a [7:0], b [7:0], c [7:0], d [7:0], e [7:0]*, signals *init* (initial installation), *ld* (loading), *clk*, and from an output are removed values of a median *out_m [7:0]* and a signal of interruption *int*. Further vectors, i.e. multidigit signals and variables will be designated, as well as in the text of the VHDL-description after the announcement of ports and signals, without the directive of quantity of digits in square brackets.

Process of designing consists in the description of functioning of a "black" box. We shall designate registers in which input data will enter, accordingly: *xa [7:0], xb [7:0], xc [7:0], xd [7:0], xe [7:0].*

Loading of input data is made at initialization of a signal *ld* and can be described by expression:

```
 if ld = ' 1 ' then xa <=a; xb <=b; xc <=c; xd <=d; xe <=e;
end if;
```

(If the signal *ld* is equal to *'1'* then signals *xa, xb, xc, xd* and *xe* values *a, b, c, d, e* are appropriated, respectively).

After record of initial data into registers it is possible to spend comparison of each entrance signal with other signals. The maximal number of steps in this case will be equal six: on the first step (we shall designate it as status S1) is made record of input data, on other five steps (S2, S3, S4, S5 and S6) – comparison of signals among themselves, summation of results of comparison, comparison of the received sums with a constant and formation of a signal of the enable of record of a code of a median into the output register. With the purpose of reduction of the hardware resources demanded for connection of compared signals to comparators, on each step parallel shift of data in registers is made, i.e. data from the register *xb* correspond in the register *xa*, from *xc* in *xb*, from *xd* in *xc*, from *xe* in *xd*:

```
a <=xb; xb <=xc; xc <=xd; xd <=xe;
```

Shift, as well as other assignment operations, is made at switching clocked signal *clk* from a status *' 0 '* in a status *' 1 '*.

For fixing of comparison results with the purpose of their further processing we shall enter variables of integer type *Integer* with area of values from 0 up to 1: *ab, ac, ad* and *ae* which are valid in a current status, and signals *ba, ca, da, ea, cb, db, eb, dc, ec* and *ed* the same type, valid for all time of process. Then the comparison operations which are carried out in various statuses can be presented in the form of:

S2 – comparison of the code containing in the register *xa* with codes, containing in registers *xb, xc, xd* and *xe* (comparison of a vector *a* with vectors *b, c, d* and *e*):

```
if xa> =xb then ab: = 1; ba <=0; else ab: = 0; ba <=1;
end if;
if xa> =xc then ac: = 1; ca <=0; else ac: = 0; ca <=1;
end if;
if xa> =xd then ad: = 1; da <=0; else ad: = 0; da <=1;
end if;
if xa> =xe then ae: = 1; ea <=0; else ae: = 0; ea <=1;
end if;
```

(if the vector *a* more or is equal *b, c, d, e* then variables *ab*, *ac*, *ad*, *ae* value *1*, and to signals *ba*, *ca*, *da*, *ea* – *0* is appropriated, otherwise *ab*, *ac*, *ad*, *ae* is appropriated *0*, and *ba*, *ca*, *da*, *ea* – *1*).

S3 – comparison of the code containing in the register *xa* with codes, containing in registers *xb*, *xd* and *xd* (comparison of a vector *b* with vectors *c*, *d* and *e*):

```
if xa> =xb then ab: = 1; cb <=0; else ab: = 0; cb <=1;
end if;
if xa> =xc then ac: = 1; db <=0; else ac: = 0; db <=1;
end if;
if xa> =xd then ad: = 1; eb <=0; else ad: = 0; eb <=1;
end if;
```

S4 – comparison of the code containing in the register *xa* with codes, containing in registers *xb* and *xc* (comparison of a vector *c* with vectors *d* and *e*):

```
if xa> =xb then ab: = 1; dc <=0; else ab: = 0; dc <=1;
end if;
if xa> =xc then ac: = 1; ec <=0; else ac: = 0; ec <=1;
end if;
```

S5 – comparison of the code containing in the register *xa* with a code, containing in the register *xb* (comparison of a vector *d* with a vector *e*):

```
if xa> =xb then ab: = 1; ed <=0; else ab: = 0; ed <=1;
end if;
```

After comparison of codes summation of results of comparison is made, thus for record of result of summation the variable of integer type *yz* with area of values from 0 up to 4 is entered:

```
S2: yz: = (ab+ac) + (ad+ae);
S3: yz: = (ab+ac) + (ad+ba);
S4: yz: = (ab+ac) + (ca+cb);
S5: yz: = (ab+da) + (db+dc);
S1: yz: = (ea+eb) + (ec+ed);
```

Final operation is comparison of result of summation with a constant (in this case 2), record, in case of the equality, current value of the code which is being the register *xa*, into output register, transition in status S1 and formation of a signal of interruption *int*:

```
 if yz=2 then out_m <=xa;
else xa <=xb; xb <=xc; xc <=xd; xd <=xe;
end if; int <= ' 1 ';
```

At an inequality of result of summation to a constant shift of data in registers is made and transition to a following status, and a signal *int* is appropriated value *'0'*. The full description of the project can be executed by means of State Editor as flowgraph.

Example of parallel realization of algorithm.
For realization of operations of comparison and calculation of results variables of integer type with area of values from 0 up to 4 are entered: *xa*, *xb*, *xc*, *xd* and *xe*. Each of variables defines a place of input code in sorted sequence and is formed as follows:

```
if (a> =d) then xa: = xa+1; else xd: = xd+1; end if;
if (a> =e) then xa: = xa+1; else xe: = xe+1; end if;
if (b> =c) then xb: = xb+1; else xc: = xc+1; end if;
 if (b> =d) then xb: = xb+1; else xd: = xd+1; end if;
 if (b> =e) then xb: = xb+1; else xe: = xe+1; end if;
 if (c> =d) then xc: = xc+1; else xd: = xd+1; end if;
 if (c> =e) then xc: = xc+1; else xe: = xe+1; end if;
 if (d> =e) then xd: = xd+1; else xe: = xe+1; end if;
```

At presence of a signal *clk* and a condition of equality any from variables to a constant (number 2), record in the output register of the input code corresponding the given variable is made:

```
  if (clk'event and clk = ' 1 ') then
  if (xa=2) then out_m <=a;
  elsif (xb=2) then out_m <=b;
  elsif (xc=2) then out_m <=c;
  elsif (xd=2) then out_m <=d;
  elsif (xe=2) then out_m <=e;
  end if; end if;
 if (a> =b) then xa: = xa+1; else xb: = xb+1; end if;
if (a> =c) then xa: = xa+1; else xc: = xc+1; end if;
```

Description MF for nine 8-digit pixels is similarly carried out.

On examples descriptions of serial-parallel and parallel realizations of algorithms in VHDL language are presented. Results of synthesis and realization of projects in crystal XCV50CS144–6, received with use of tool means FPGA Express and programs of placement and routing of system Xilinx Foundation Series, are presented in Table 3.2.

In realization of variant SL the maximal time of definition of a median T_1 to equally product tN_1, where t – the minimal period clk, $N_1 = [(n^2 - n)/2 + 1]$ – the number of statuses equal (for $n = 5$) in view of loading eleven. If a median is the input code x_1 it will be certain on the fifth step (the first step corresponds to loading), x_2 – on the eighth step, x_3 – on the tenth step, x_4 – on the eleventh and x_5 – on the first step of a following cycle corresponding loading of a new portion of input data. Thus, the size T_1 will be within the range of from $(n-1)t$ up to $[(n^2 - n)/2 + 1]t$.

Table 3.2 Results of synthesis

Type of realization of algorithm	Quantity of 8-digit pixels	Hardware resources (quantity of Slices)	Min. period [ns] / max. freq. [MHz]	Max time of definition of a median [ns]
SL	5	73	12 / 87	132
SP	5	62	13 / 77	65
PR	5	87	15 / 67	15
PR	9	317	40 / 25	40

Maximal time of definition of a median T_2 for realization of a variant of software to equally product tN_2, where $N_2 = n$ – the number of statuses equal (for $n = 5$) in view of loading 5. Definition of a median occurs on a step which number corresponds to number of a code in entrance sequence plus 1. If a median is the input code x_1 it will be certain on the second step (the first step corresponds to loading), x_2 – on the third step, etc. Time T_2 will be within the limits of from t up to nt. At fixing the fact of definition of a median return to an initial status – S1 is carried out and the signal of interruption of process is formed.

At realization of variant PR time of definition of a median T_3 is size of a constant (t) and is equal 15ns / 40ns accordingly for five / nine pixels.

Thus, variant PR (for $n = 5$) uses approximately in 1,4 (in comparison from software) and accordingly in 1,2 times (in comparison with SL) more slices in a crystal (an estimation of hardware expenses), possessing thus approximately in 4,5 (in comparison from software) and accordingly in 9 (in comparison with SL) time greater speed.

3.5.4 Hemming Adder Realization

Let's consider following variants of realization of Hemming adder:

HA1 is realization of the multilevel combinational scheme on the basis of logic elements AND, XOR by means of Schematic editor.

HA2 realizes the adder, using a tree chart of the adder on which top level in pairs weighed sums two components are formed, and further on the basis of standard schemes of adders - result of the weighed sum. All elements HA2 are created by means of system Core Generator as functionally completed blocks and, eventually by means of the schematic editor the resulting scheme is formed.

HA3 is realized by the behavioral description by VHDL language which is resulted below.

```
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.std_logic_unsigned.all;
entity hamm_v is
           generic (N:Integer: = 32);
                 port (
                 a: in STD_LOGIC_VECTOR (N-1 downto 0);
                 y: out INTEGER range 0 to N
                 );
end hamm_v;
architecture Behavioral of hamm_v is
begin
process (a)
variable x: integer range 0 to N;
begin
                           x: = 0;
                           for I in 0 to N-1 loop
                           if a (I) = ' 1 ' then
        x: = x+1;
                           end if;
                           end loop;
                           y <=x;
        end process;
end Behavioral;
```

Synthesis HA3 is executed by means of FPGA Express (Synopsys).

In tab. 3.3 comparative characteristics of devices (HA1, HA2, HA3), carrying out calculation of Hemming distance for 4–, 8– and 16–bit numbers realized in a crystal Virtex series are resulted.

The offered realizations of algorithms differ hardware resources (hardware resources are understood as dimension of a processing field or the logic capacity of a crystal defined by quantity Slices) and speed.

Table 3.3 Comparative characteristics of devices HA1, HA2, HA3

Variant of realization	Speed (the period [ns] / frequency [MHz])			Hardware resources (quantity Slices)		
	n = 4	n = 8	n = 16	n = 4	n = 8	n = 16
HA1	4,4 / 227,2	8,1 / 123,4	11,9 / 84	4	10	38
HA2	7,8 / 128,2	10,2 / 98	13,4 / 74,6	4	10	22
HA3	3,6 / 277,8	5,5 / 181,8	12,7 / 78,8	2	5	22

On the basis of the received estimations it is possible to draw following conclusions. Variant HA3 has the best parameters of estimations on hardware resources for any word length (for $n = 16$ an resources coincide with variant HA2), on speed slightly conceding only to variant HA1 for $n = 16$. HA2 has no advantages before other realizations, confirming known regulations about volume, that the complex system from optimum components not necessarily is optimum in aggregate. However the basic advantage of variant HA3 is that the presented behavioral description, is the parametrical description of Hemming adder, i.e. universal (for any word length). The task of parameter *(N)* in the description *(generic)* defines word length of the synthesized adder.

3.6 Verification of Projects by Means of Stands

Let's consider the description process project on an example of the median filter which block–diagram (Fig. 3.6), contains the block from five 8-bit registers, outputs of each of which are connected to inputs of other register and inputs of the device for definition of a median.

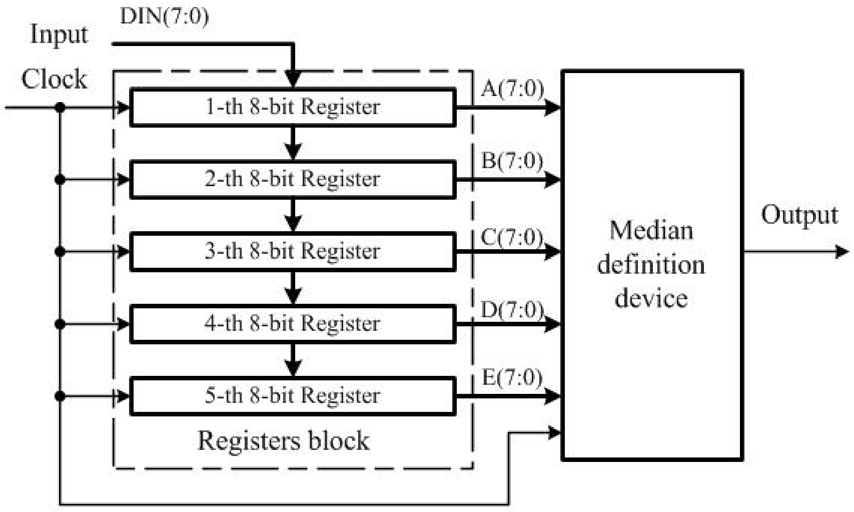

Fig. 3.6 Block–diagram of the median filter

On an input of the filter (n = 5, m = 8) the file of 8–bit codes, for example from output ADC consistently acts. With signal Clock codes move from 1–st register in 2–nd, from 2–nd in 3–rd, etc.

Simultaneously with everyone Clock transfer of codes from all 5 registers on inputs of the device for a median definition is carried out. Definition of a median is carried out on 5 pixels (codes) according to algorithm and the VHDL–description, and modified due to inclusion in the device for median definition a of four registers blocks for realization of a conveyor mode. The project of the median filter contains, thus, the scheme, being top level of hierarchy of the project, and two modules: the registers block and the device for the median definition, executed as VHDL–descriptions.

In Fig. 3.7 the scheme of the filter synthesized by means of the schematic editor, registers block is presented in the form of the text of VHDL–description, is resulted. Description of the registers block as text is much more compact and easier, than the description of this module by means of the schematic editor.

In Fig. 3.7 this module is presented in the form of an environment of a macrocell from the user library. VHDL description of the median filter:

```
library IEEE; -- library declaration
use IEEE.STD_LOGIC_1164. ALL; -- using declaration
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
entity reg_s is -- object declaration
   Port (din: in std_logic_vector (7 downto 0); -- port declaration
        clk: in std_logic;
        dout_a: out std_logic_vector (7 downto 0);
        dout_b: out std_logic_vector (7 downto 0);
        dout_c: out std_logic_vector (7 downto 0);
        dout_d: out std_logic_vector (7 downto 0);
        dout_e: out std_logic_vector (7 downto 0));
end reg_s;
architecture Behavioral of reg_s is – architecture declaration
signal va, vb, vc, vd, ve: std_logic_vector (7 downto 0); -- signal declaration
begin
```

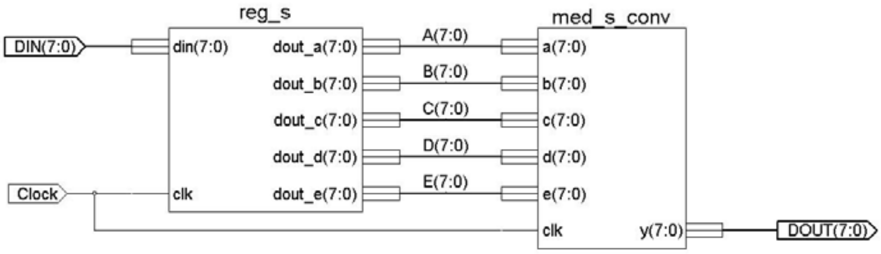

Fig. 3.7 Functional schema of the median filter

At the description of this module by means of the Schematic editor it would be necessary to execute the scheme containing five 8-bit registers FD8CE where register FD8CE would be presented by the scheme of lower level of the hierarchy consisting of eight triggers FDCE of system library.

The considered example of the median filter has been synthesized, placed and routing in crystal XC2S150–5PQ208 (Spartan–II series).

At realization the project has borrowed 9 % of resources of a crystal, i.e. 169 of 1728 Slices, one global buffer, 16 pinouts (IOB). The maximal clock frequency is equal 95 MHz.

Project Verification

The basic tool of verification of the project is the modeling system of Model Technology ModelSim, Xilinx Edition (MXE II). One of laborious processes of verification is a file processing of input influences on model-based object. For simple projects can be used HDL Bencher – the graphic interface for creation of input influences in the form of sequences of the impulses set by the user (Waveform). For projects where input influences are a product of complex logic transformations or a codes file, creation of the virtual stand or stands for verification of the project is expedient. Such stand can be executed in the form of the subproject included in a separate branch of a tree of hierarchy of the developed project. Further the test (HDL Test Bench) is described in the form of structure where the stand and the project are presented in the form of the interconnected components.

Let's consider as an example development of the stand for functional check of the median filter which description has been resulted above (see Fig. 3.6).

Let the filter makes "clearing" the signal consisting of a "useful" signal of the sine wave form and formed "noise". Thus, the stand can consist of the shaper of a signal of the sine wave form, the generator of random numbers and the multiplexer which is carrying out transfer of a code from an output of the shaper or the generator on an input of the filter (Fig. 3.8).

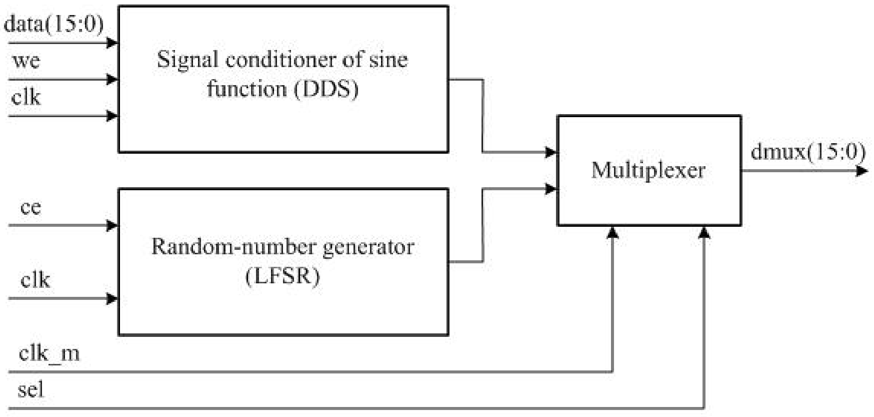
data(15:0)
we
clk
Signal conditioner of sine function (DDS)
ce
clk
Random-number generator (LFSR)
clk_m
sel
Multiplexer
dmux(15:0)

Fig. 3.8 Block–diagram of the stand for functional check of the median filter

As the shaper of a signal of the sine wave form it is used IP–Core – Direct Digital Synthesizer (DDS), as the generator of random numbers – the Linear Feedback Shift Register (LFSR).

In Fig. 3.9 the timing diagram of the filtration process, received is resulted at functional modeling of the median filter with using of the specified stand.

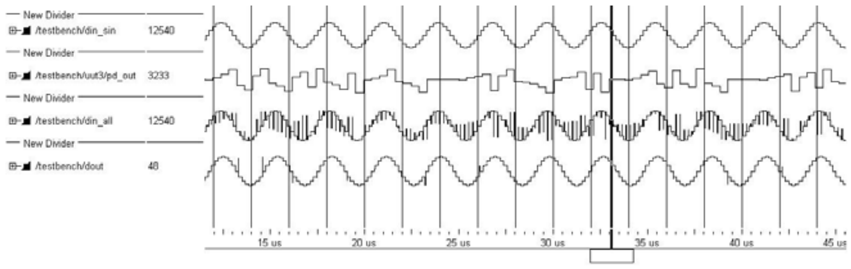

Fig. 3.9 The timing diagram of a filtration process:
a – signal on an output of the shaper of the sine wave form;
b – signal on an output of the generator of random numbers;
c – signal on an input of the median filter;
d – "cleared" signal on an output of the median filter.

3.7 Reconfigurable Processors

The typical structure of reconfigurable processor (RP) allows the developer (user) to realize any algorithm, i.e. to change structure depending on a carried out problem (the set algorithm). Last can be broken into the fragments which are carried out consistently on fixed hardware that leads to the general economy of hardware, thus complexity of fragments of algorithm is defined only by logic capacity of a crystal FPGA. Presence of set of functional processing fields (FPF) allows is hardware to realize parallel data processing, and set of configuration files – conveyor programming of the structure realizing fragments of algorithm.

Reconfigurable processors have FPF the set dimension which is configured for performance of the set algorithm or its part, providing, thus, optimum realization of this algorithm, both under time characteristics, and on hardware expenses. At the conveyor mechanism of realization of algorithm additional matrixes are entered into structure RP. Structure RP is presented in Fig. 3.10 and contains S matrixes FPF, the channel of input-output (CIO) for connection to the standard Bus of a Host-computer, a memory of a configuration files (MCF), the data RAM, the controller (CT), data bus (DB) and control bus (CB). The conveyor mechanism assumes loading a configuration file in the next matrix in parallel with data processing in a current matrix.

The format of a configuration file is standard for FPGA and contains the information about of a configuration matrix, i.e. forms the corresponding basic electric scheme realizing set algorithm. Matrix FPF represents a matrix of universal elements which under control of a configuration file F_γ direct function is appointed and the structure of communications between them is formed. Configuration files F_{γ} enter the name in matrix FPF from a memory of configuration files under control of CT.

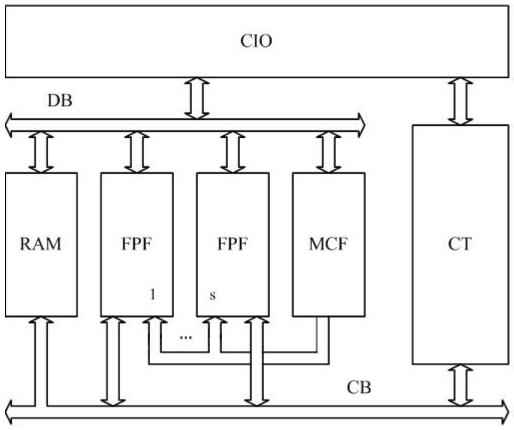

Fig. 3.10 Structure of reconfigurable processor

In matrix FPF on data bus details from the RAM or external entrance data through CIO can act. Results of processing from matrix FPF can be transferred in channel CIO as external target data or in the RAM as intermediate results. The set of files of a configuration $F = \{F_{\gamma}\}$ enters the name in the MCF through channel CIO under control of CT.

Initialization of system consists of three stages: record of set of files of a configuration F in the MCF; loading of files of a configuration F_{γ} in FPF from the memory of configuration files; functioning of system – realization of algorithm.

Procedure of data processing is carried out as follows. In a command number (α) matrixes FPF is underlined, and CT forms a signal initializing corresponding matrix FPF. Then loading of a corresponding file of a configuration in next matrix FPF is carried out. After end of data processing by α-th matrix results of data processing enter the name in the RAM and serve as intermediate (initial) data for $(\alpha+1)$-th matrix. Upon termination of work of algorithm with given FPF (the termination of the microprogramma) is formed interruption which acts on operating input CT where its processing is carried out.

3.8 Conclusions

The received results have allowed to raise efficiency display of initial problems and algorithms to architecture and structure of projected PLD-based devices and systems by criteria « speed – complexity of realization » on the basis of the developed formalized techniques of construction and dynamic reorganization of their architecture and structure, proceeding from properties of sold algorithms, and also logic, constructive and technological features PLD, and tool means of their designing.

As a result of the executed analysis of evolution, tendencies of development and technology of realization of a new class of components computer engineering – programmable logic devices it is certain, that PLD opportunities of construction on their basis of devices and the systems possessing properties of the reconfigurability, providing give adaptation to a wide spectrum of problems and reception of high characteristics of projected devices and systems.

Principles of construction and functioning of a new class of computers and systems with reconfigurable the architecture are developed, differing from traditional (von Neumann type) properties of high dynamic reorganization, multilevel and parallelism of data processing that functional means of computer engineering for any algorithms allow the developer (user) to create, providing thus an opportunity of structural adaptation, including in real time, according to a solved problem (algorithm) and also to duplicate them for a wide range of developers, reducing process of designing of digital devices to a choice from library of optimum structure by criteria «speed – complexity of realization» with adjustment of corresponding parameters.

The known logical-information method of designing reconfigurable devices and systems which basic difference became orientation to functionalities PLD is modified. In the offered kind it allows to operate with any quantity of levels of programming, to define optimum quantity of such levels and to synthesize the optimum structure of the device represented by multilevel hierarchical system with unlimited number of levels on a class of criteria « speed – complexity of realization ».

The new class of computing structures – adaptive logic networks (ALN), principles and techniques of their construction and functioning are offered. It is shown, how for base set of structures ALN and training samples, the binary vectors set by set, using polynomial representation, which factors are set by means of Adamar matrix, it is possible to receive analytically set of logic functions (functional adjustment) components ALN at the functional restrictions preliminary certain also analytical by that allows, passing process of direct synthesis to execute predesign estimations of a realizability of developed devices. Process of designing consists in correct display of entrance set of data in target set of data and is reduced to is formal-analytical procedure of decomposition with use of preliminary received functional restrictions. The offered device effectively supports process of adaptation ALN on classes of problems which are reduced to procedure of classification, including problems of natural classification.

A number of structural realizations ALN is offered: in the form of "triangular", "trapezoidal" and "rectangular" matrixes which covers a wide class of problems. Process of adjustment of matrixes is reduced to definition of types of logic functions elementary a component and structures of communication from the limited set is set. The offered structures differ on capacity sold Boolean functions and to hardware expenses, are accompanied received analytical by asymptotic by estimations of complexity (depending on word length of entrance binary vectors) and capacities of target set of binary vectors.

The open library of functional devices which structure can extend and be oriented to problem is developed. In particular, it is devices: definitions of a median with time step by step conveyor processing of input data; memory-based sorting of data; adders Hemming (for any word length); multiplication of matrixes; multipliers with a floating point (standard IEEE-754), etc.

The base structure of reconfigurable processor with set of functional fields which allows to focus functionally it on an any class of problems (algorithms) is developed, supporting, in particular, parallel, conveyor and in parallel-conveyor data processing. The developed processor is a basis for construction of a lot of computing systems of high complexity, productivity and survivability.

References

[1] Estrin, G., Turn, R.: Parallel processing in a restructurable computer. IEEE Transaction on Electronic Computers EC 12(6), 747–755 (1963)

[2] Athanas, P.M., Schewel, J., McHenry, J.T., James–Roxby, P.B.: Reconfigurable Technology: FPGAs and Reconfigurable Processors for Computing and Communication. In: SPIE International Society for Optical Engineering, p. 174 (2001)

[3] Villasenor, J., Mangione–Smith, W.H.: Configurable Computing, `http://www.vcc.com`

[4] Lord, E., Cantle, A.J., Dr Devlin, M., Shand, D.: COTS Platform for the Development of Re-configurable Processing in Aerospace Systems. White paper, `http://www.nallatech.com`

[5] Schewel, J.: Hardware / Software Co–Design System using Configurable Computing Technology, `http://www.vcc.com`

[6] Palagin, A.V., Opanasenko, V.N.: Reconfigurable computing technology. Cybernetics and Systems Analysis 43(5), 675–686 (2007)

[7] Palagin, A.V., Opanasenko, V.N., Sakharin, V.G.: Features of Digital Devices Design of Modern PLD of the Xilinx Incorporation. Journal of Automation and Information Sciences 33(3), 80–89 (2001)

[8] Cosoroaba, A., Rivoallon, F.: Achieving Higher System Performance with the Virtex–5 Family of FPGAs. Xilinx Inc. White Paper WP245 (v1.1) May 17 (2006), `http://www.xilinx.com`

[9] Chang, C., Wawrzynek, J., Brodersen, R.W.: BEE2: A High–End Reconfigurable Computing System. IEEE Design and Test of Computers 22(2), 114–125 (2005)

[10] Bruck, J., Blaum, M.: Neural networks, error–correcting codes, and polynomials over the binary n–cube. IEEE Transactions on information theory 35(5), 976–987 (1989)

[11] Palagin, A.V., Opanasenko, V.N., Chigirik, L.G.: Synthesis of a Hamming network on a basis of programmable logic integrated circuits. Engineering Simulation 13, 651–666 (1996)

[12] Astola, J., Haavisto, P., Neuvo, Y.: Vector median filters. Proc. of the IEEE 78(4), 678–689 (1990)