ISSN 1063-7397, Russian Microelectronics, 2021, Vol. 50, No. 6, pp. 426–438. © Pleiades Publishing, Ltd., 2021.
Russian Text © The Author(s), 2021, published in Mikroelektronika, 2021, Vol. 50, No. 6, pp. 467–480.

Designing FPGAs and Reconfigurable SoCs Using Methods of Program Analysis and Prototyping

V. I. Enns[a], *, S. V. Gavrilov[b], **, V. M. Khvatov[b], *, and V. G. Kurbatov[a]**

[a] *AO Molecular Electronics Research Institute (AO MERI), Zelenograd, Moscow, 124460 Russia*

[b] *Institute for Design Problems in Microelectronics, Russian Academy of Sciences (IDPM RAS), Zelenograd, Moscow, 124365 Russia*

**e-mail: venns@niime.ru*

***e-mail: sergey_g@ippm.ru*

****e-mail: khvatov_v@ippm.ru*

Received May 14, 2021; revised June 8, 2021; accepted June 15, 2021

Abstract—This paper describes approaches and methods for software circuit prototyping of field-programmable gate arrays (FPGAs) and reconfigurable systems-on-a-chip (RSoC). Software circuit prototyping is a new stage in the design flow for FPGAs and reconfigurable SoCs in contrast to the classical FPGA-based prototyping using ready-made FPGA chips. It allows to evaluate the efficiency of the user circuit design implementation and select the basic chip architecture before its tape-out because of the computer-aided design (CAD) tools promptly adapting to any changes in the structure, circuitry and layout of a basic chip. The flexible and dynamic software customization to maintain the required FPGA or RSoC architecture is provided by the developed formalized description of the basic circuit, which is used in CAD and is represented in this paper. This formalized description can be used both for the analysis of the basic chip and for the analysis of a user circuit design implemented on an FPGA or RSoC.

DOI: 10.1134/S106373972106007X

INTRODUCTION

Designing field programmable gate arrays (FPGAs) or reconfigurable systems-on-a-chip (RSoC) is a complex process that requires considerable time to select the parameters of a circuit and its architecture, analyze routing capabilities, and simulate circuit components [1–4]. Despite the fact most of the design flow stages are automated, the routing evaluation of the user circuit design and the search for the developed architecture's bottlenecks are performed by humans. The approach without verification of the architecture could lead to errors that could be found only after the layout design stage of the basic chip. Errors at this stage are unacceptable in a time-limited design and manufacturing process.

Methods of software circuit analysis and architecture evaluation can significantly simplify and speed up the design process of an FPGA or RSoC basic chip. The existing evaluation methods based on the execution of the full design flow [5–9] use a simplified description of a basic chip, which does not allow to accurately evaluate its architecture, its specific details, or weaknesses. Also, the available methods require a circuit description in a specialized format. This poses an additional task of studying new methods of presenting the developed circuit schematic view for the chip architect.

In the proposed approach to assess the FPGA and SoC's architecture, the Circuit Design Language (CDL) [10] is applied for a basic chip description. This format is often used by integrated circuit (IC) designers and is utilized by computer-aided design (CAD) systems produced by Cadence, Synopsys, and Mentor Graphics. The circuit description in this format is automatically generated from the graphical representation in the schematic editor of the manufacturers mentioned above. The CDL support together with the developed formalization of the circuit representation in CAD provides a flexible and quick customization of the software considering the corresponding changes in the structure, circuitry, and layout of the developed heterogeneous SoC or FPGA. It also allows us to evaluate the efficiency of various user circuit design implementations and the architecture of a basic chip.

The developed method, which includes a quick customization of the CAD software, that takes into account the changes in the basic chip architecture, and evaluation of the architecture's efficiency is a new stage in the reconfigurable or programmable logic design flow, called *software circuit prototyping*. Software circuit prototyping means verification and evalu-

[1] The text was translated by the authors.

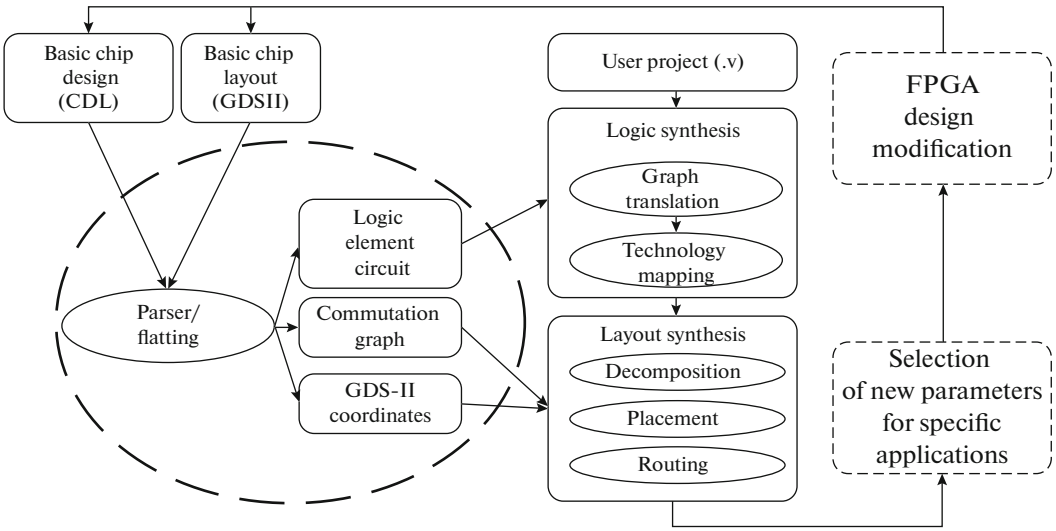

Fig. 1. Software circuit prototyping stages.

ation of the basic chip containing FPGA elements before its tape-out, in contrast to classical prototyping, which means verification of the system on a chip or its individual IP-cores based on the manufactured FPGA chip [11, 12].

SOFTWARE CIRCUIT PROTOTYPING METHOD

The software prototyping method proposed in this study consists of several stages (Fig. 1).

(1) The first step is the selection and design of the underlying architecture. It is the base for prototyping and further modifications. The architecture can be either unique or selected from a variety of existing solutions that differ in the structure of routing resources [13, 14] (island style and hierarchical FPGA architectures), the structure of a logic element (LE) and a logic array blocks (LABs) (LE in FPGAs by Altera [15], a configuration logic block in FPGAs by Xilinx [16], a versatile logic cell in FPGAs by Microsemi [17]). The final circuit architecture from all the variety available can be selected using physical constraints such as the package size or the chip area required to fit the configuration memory. Also, restrictions are imposed based on the requirements set by specific user projects, which are denoted in terms of the volume of programmable logic, routing capabilities of the basic chip, and the variety of IP-cores.

(2) At the second stage, information about the developed circuit is transferred to the CAD database. Initially the CAD system processes and analyzes the schematic view of the RSoC or FPGA circuit in the CDL file format and its layout in the GDS II file format with the help of specialized software, the so-called parser [18]. The program structure automatically adapts to the FPGA architecture by processing these files. The program generates a routing resources graph, coordinates of logic blocks, and a memory card, based on which the firmware vector is formed. The opportunity to automatically customize CAD tools to any architecture allows RSoC and FPGA developers to evaluate the routability of the circuit and to find the architecture's weaknesses in advance. It also enables CAD developers to debug the software for the future architecture according to the customer's needs. This feature makes the development process and the final result much more efficient.

(3) At the next stage, a complete design flow of the user circuit is performed, including logic and layout synthesis [19]. Logic synthesis consists of graph translation and technology mapping into the basis of the target FPGA or RSoC chip [20, 21]. Layout synthesis, in turn, includes the netlist decomposition into separate groups or clusters [22], placement of logic elements on the legal positions of the FPGA matrix [23], and routing connections between LEs and I/Os using routing resources embedded in the architecture [24].

(4) The final stage of software circuit prototyping is the analysis of the results. The new architecture parameters are selected and the corresponding changes are made to the schematic view of the basic chip based on the performed analysis. Software circuit prototyping is an iterative process; thus, the stage of changes to the circuit's design only completes one iteration of the selection of the required architecture. The prototyping process can be considered complete when two conditions are met. The first condition is that the prototyping results meet all the specified requirements and constraints. The second condition is that the full design flow has been successfully completed for the user circuit design set. If these conditions are not met, the basic chip architecture changes and the process is repeated pending a positive result.

The following section III shows the features of a basic chip presentation and the user circuit design description in CAD to perform software circuit prototyping. In subsection III.a, the stage of loading a basic chip into the CAD system is considered in more detail. Also, the developed formalized view of the FPGA or RSoC schematic view is shown. Subsection III.b presents the features of analysis and processing of the FPGA and SoC layout in CAD. Section IV contains the practical results of using the developed method for software prototyping of the basic FPGA architecture, and also describes the architecture changing parameters and characteristics, based on which the obtained prototypes were compared.

FEATURES OF REPRESENTATION OF A BASIC CHIP AND THE USER CIRCUIT DESIGN IN CAD TO PERFORM SOFTWARE CIRCUIT PROTOTYPING

Features of the Representation of the Basic Chip and the User Circuit Design

Prompt adjustment of the design and circuitry of a basic chip to new needs from the end user, as well as the fast adjustment of the CAD system for corresponding changes in the design, circuitry, and layout of a basic chip, is provided by formalizing the correspondences between the elements of the basic design of a reconfigurable or programmable heterogeneous SoC or FPGA from the manufacturer (base) and user circuit design from the end customer.

To load the required information into CAD, the circuit of a basic chip (reconfigurable or programmable heterogeneous SoC or FPGA) is presented as a description in the CDL format, and the user circuit design is presented as a flat netlist in the Verilog language.

During processing the basic chip circuitry, its hierarchical description is defined in the CAD system as an ordered triple:

$$\Pi = (S, L, s_m)\text{—is a hierarchical project description,} \quad (1)$$

where $S = \{s_i,\ i = 1,\ldots,|S|\}$ is the set of circuits in a hierarchical project description;

$L \subset S$ is a basis or subset of library subcircuits for the current design level (or stage);

$s_m \in S$, $s_m \notin L$, is the main circuit or top-level circuit.

At the same time, each of the schematic views in the project hierarchy is defined as follows:

$$\forall s \in S\colon s = (\mu(s), E(s), N(s), P(s), C(s)), \quad (2)$$

where $\mu(s)$ is a unique circuit name (character string);

$E(s) = \{e_i, i = 1,\ldots,|E(s)|\}$ is the set of elements in the circuit;

$N(s) = \{n_i, i = 1,\ldots,|N(s)|\}$ is the set of nets (nodes) in the circuit;

$P(s) = \{p_i, i = 1,\ldots,|P(s)|\}$ is the set of external pins (contacts) of the circuit;

$C(s) = \{c_i, i = 1,\ldots,|C(s)|\}$ is the set of connections (commutations) of the circuit.

The set of elements is characterized by the following components:

$$\forall e \in E(s)\colon e = (\mu(e), m(e), P(e)), \quad (3)$$

where $\mu(e)$ is a unique element name (character string);

$m(e) \in S$ is an element model represented in the hierarchical description by a subcircuit of the next (lower) hierarchy level;

$P(e) = \{p_i, i = 1,\ldots,|P(e)|\}$ is the set of element pins that match the composition of the set of element model external pins (a one-to-one correspondence between them is assumed):

$$P(e) \leftrightarrow P(m(e));\ \ |P(e)| = |P(m(e))|. \quad (4)$$

The set of external circuit pins is characterized by the following components:

$$\forall p \in P(s)\colon p = (\mu(p), \tau(p)), \quad (5)$$

where $\mu(p)$ is a unique pin name;

$\tau(p) \in \{\tau_{inp}, \tau_{out}, \tau_{bi}\}$ is a pin type: input, output, and bidirectional.

The set of circuit nets (nodes) is characterized by a name and a set of net connections:

$$\forall n \in N(s)\colon n = \mu(n), \quad (6)$$

where $\mu(n)$ is a net (node) name, (character string);

$C(s)$ is a set of connections in a circuit, defined as a subset of such pairs

$$C(s) = \left\{(p,n)\colon p \in \left(\bigcup_{i=1,\ldots,|E(s)|} P(e_i) \cup P(s)\right),\ n \in N(s)\right\} \quad (7)$$

that the net is unique or does not exist at all for any contact:

$$\forall p \in \left\{ \bigcup_{i=1,\ldots,|E(s)|} P(e_i) \cup P(s) \right\}: \\ (\exists! n \in N(s): (p,n) \in C(s)) \vee \\ \vee (\forall n \in N(s): (p,n) \notin C(s)). \tag{8}$$

In other words, the set of circuit connections is defined as a unique mapping:

$$C^*(s) = \left\{ \left(\bigcup_{i=1,\ldots,|E(s)|} P(e_i) \cup P(s) \right) \to (N(s) \cup \emptyset) \right\}. \tag{9}$$

At the same time, the inverse mapping determines the actual connections list of the net and cannot be injective; i.e., the number of connections for each net must be at least two; otherwise the net is considered erroneous or false:

$$C^{*-1}(s) = \left\{ N(s) \to \left(\bigcup_{i=1,\ldots,|E(s)|} P(e_i) \cup P(s) \right) \right\} \\ \forall n \in N(s): \left| \{(p,n): (p,n) \in C(s)\} \right| \geq 2. \tag{10}$$

A net can have only one external pin as a rule:

$$\forall n \in \\ \in N(s): \left| \{(p,n): (p,n) \in C(s) \wedge p \in P(s)\} \right| \leq 1. \tag{11}$$

The difference between the formal description of the basic design and the formal description of the user circuit design is that the external pin name in the schematic view is the same as the name of the net connected to it:

$$\forall n \in N(s), \\ \forall p \in P(s): (p,n) \in C(s) \Rightarrow \mu(n) = \mu(p). \tag{12}$$

At this design stage, the circuits of the basic library level are "black boxes"; i.e., they do not contain internal data:

$$\forall s \in L: E(s) = \emptyset, \quad N(s) = \emptyset, \quad C(s) = \emptyset. \tag{13}$$

In this case, the lower level subcircuits are modeled based on the built-in models and the description of the black boxes can be hidden from the external user. For example, at the schematic design level, the basic library level includes transistors, capacitances, resistances, and inductances.

The hierarchical description of the underlying chip is converted into a corresponding flat representation in the CAD system by a recursive flatting procedure. For the given project $\Pi = (S, L, s_m)$ only those subcircuits that are actually used in s_m are saved in the flat view.

Let us denote by $\varphi(s, s_t)$ a logical function defined on the Cartesian product $S \times S$ taking value 1 if and only if s is actually used in s_t:

$$\varphi: S \times S \to \mathfrak{B}; \quad \mathfrak{B} = \{0,1\}; \quad \varphi(s, s_t) = \left((s = s_t) \vee \left(\bigvee_{e \in E(s_t)} \varphi(s, m(e)) \right) \right), \tag{14}$$

i.e., $\varphi(s, s_t) = 1$, if $(s = s_t)$ or $\exists e \in E(s_t)$: $\varphi(s, m(e)) = 1$.

The "flat representation" $\Pi_f(\Pi) = (S_f, L_f, s_f)$ for the given project $\Pi = (S, L, s_m)$ is built using the following rules:

$L_f = \{s: (s \in L) \wedge \varphi(s, s_m)\}$;
$S_f = \{s_f \cup L_f\}$;
$s_f = (\mu(s_f), E(s_f), N(s_f), P(s_f), C(s_f))$; where
$\mu(s_f) = \mu(s_m)$;
$P(s_f) \leftrightarrow P(s_m)$.

Element names $\mu(e), e \in E(s_f)$, and net names $\mu(n), n \in N(s_f)$, in a flat representation are unique and contain information about the element names of higher levels of the hierarchy, which include subcircuits containing the considered element, before the hierarchy is expanded.

A flat view consists of the elements contained in the basic design library. The following elements types can be identified in the library: logic elements L_{LE}, peripheral (IO) input/output elements L_{IO}, complex-functional macroblocks L_M or IP-cores, routing elements L_{Ro}, and other auxiliary elements L_{BB} (black boxes) that do not contain any of the listed element types L_{LE}, L_{IO}, L_M, or L_{Ro} and perform additional functions elements (for example, memory programming), not related to the mapping of elements of a user circuit design:

$$L = L_{LE} \cup L_{IO} \cup L_M \cup L_{Ro} \cup L_{BB}. \tag{15}$$

When converting a hierarchical basic design to a flat representation, the fact of the presence of elements and the actual number of elements in all listed

types and in each subcircuit of a higher level of the hierarchy is considered. The fact that subcircuit s belongs to the set of black boxes is determined by the absence of the listed types elements in it, if we denote the actual number of elements of the listed types in the given subcircuit through $\sigma_{LE}(s)$, $\sigma_{IO}(s)$, $\sigma_{M}(s)$, $\sigma_{Ro}(s)$. Moreover, this does not depend on whether the circuit has subcircuits and elements of a lower level:

$$s \in L_{BB} \equiv (\sigma_{LE}(s) + \sigma_{IO}(s) + \sigma_{M}(s) + \sigma_{Ro}(s) = 0). \quad (16)$$

At the same time, the value of each of the listed functions for counting elements of the corresponding type $T \in \{LE, IO, M, Ro\}$ can be determined recursively:

$$\sigma_T: S \to \mathcal{N}_0, \quad \mathcal{N}_0 = \mathcal{N} \cup 0;$$

$$\sigma_T(s) = \begin{cases} 1 \quad \text{when} \quad s \in L_T \\ 0 \quad \text{when} \quad s \in L \backslash L_T \\ \sum\limits_{e \in E(s)} \sigma_T(m(e)) \quad \text{when} \quad s \notin L \end{cases}. \quad (17)$$

As a result of these calculations, the conversion of a hierarchical basic design to the flat representation is limited by the level $L = L_{LE} \cup L_{IO} \cup L_M \cup L_{Ro} \cup L_{BB}$.

Elements e: $m(e) \in L_{LE} \cup L_{IO} \cup L_M$ are used to map library elements in a user design project. Elements e: $m(e) \in L_{Ro}$ are used to map nets and connections of a user design. Based on them, a graph is automatically built for solving routing problems.

Library-level circuits $s \in L_{LE} \cup L_{IO} \cup L_M = L \backslash \{L_{BB} \cup L_{Ro}\}$ are programmed by the use of the library for various functional solutions through programmable memory. The set of external pins of such circuits is defined as follows:

$$P(s) = \{p_i,\ i = 1,\ldots,|P(s)|\}, \quad s \in L_{LE} \cup L_{IO} \cup L_M \quad (18)$$

$$P(s) = P_r(s) \cup P_m(s) \cup P_s(s). \quad (19)$$

It is divided into three independent subsets with different functional purposes:

where $P_r(s)$ is a subset of signal or routing pins for connecting external signal nets using routing resources from L_{Ro};

$P_m(s)$ is a subset of programmable outputs to control various options of functional solutions;

$P_s(s)$ is a subset of service pins for connecting special signals (for example, power, ground, synchronization, and reset), the connection of which requires special processing other than connecting conventional nets or signals.

The cardinality of the set $|P_r(s)|$ determines the maximum number of element pins allowed in the user library L_u of the user project $\Pi_u = (S_u, L_u, s_{mu})$. Some of the pins $P_r(s)$ may not be used in a specific library element from L_u or may be connected to ground/power.

The cardinality of the set $|P_m(s)|$ determines the length of the programming vector for the implementation of specific functions and work modes of library elements. Due to different programming options, one instance $s \in L_{LE} \cup L_{IO} \cup L_M$ can be used for many different implementations in a user library L_u. The maximum number of implementations in the library can be $2^{|P_m(s)|}$. In particular, the number of functions for a classic LookUp Table (LUT) element [25] with n inputs is

$$2^{|P_m(s)|} = 2^{2^n}. \quad (20)$$

Thus, the formation of the library elements $s_u \in L_u$, $s_u = (\mu(s_u), \emptyset, \emptyset, P(s_u), \emptyset)$, of the user library L_u of the project $\Pi_u = (S_u, L_u, s_{mu})$ is realized by setting the following relations for the pins of the library circuits of the basic chip $P(s) = \{p_i, i = 1,\ldots,|P(s)|\}$, $s \in L_{LE} \cup L_{IO} \cup L_M$:

$$P_m(s) \to \mathcal{B}^{|P_m(s)|}, \quad \mathcal{B} = \{0,1\}; \quad P_r(s) \to P(s_u) \cup \{P_0, P_1, P_z\}, \quad (21)$$

here P_0, P_1, P_z are the notation keys for pins that have external connections to a ground, power, or an unconnected node.

Without loss of generality, it can be assumed that a user design circuit from the end customer $\Pi_u = (S_u, L_u, s_{mu})$ is specified in a flat representation, obtained as a result of automatic synthesis from an RTL description, or it is a result of the direct conversion of a hierarchical user description into a flat representation; then $S_u = L_u \cup \{s_{mu}\}$.

Let us suppose the user top circuit is $s_{mu} = (\mu(s_{mu}), E(s_{mu}), N(s_{mu}), P(s_{mu})C(s_{mu}))$. External pins of a user circuit design $p_u \in P(s_{mu})$, can be processed in two ways:

– The first way involves the assignment of peripheral elements from L_{IO}. At the same time, depending on the type of pin $\tau(p_u) \in \{\tau_{inp}, \tau_{out}, \tau_{bi}\}$ various peripheral elements modes are programmed: input, output, or bidirectional.

– The second way assumes that the peripheral elements have been selected at the stage of forming the user circuit design and the circuit pins $p_u \in P(s_{mu})$ are external interfaces for modeling.

The stage of assigning peripheral elements involves not only the selection of a specific type of peripheral element $s_{IO} \in L_{IO}$ with programming,

$$P_m(s_{IO}) \to \mathcal{B}^{|P_m(s_{IO})|}, \quad \mathcal{B} = \{0,1\}, \quad (22)$$

but also the selection of a specific instance of a peripheral element $e \in E(s_f)$. Therefore, a specific place of this element in the flat representation of a basic chip, i.e., matching (mapping) is obtained:

$$P(s_{mu}) \to \{e\colon e \in E(s_f), \\ m(e) = s_{IO}, \quad s_{IO} \in L_{IO}\}. \tag{23}$$

In this case, the procedure for assigning a specific peripheral instance and its placement can be performed both in the manual or interactive mode, and automatically.

A similar problem is solved for all internal elements of the user circuit design, both for the standard logic elements and for complex-functional macroblocks:

$$E(s_{mu}) \to \{e\colon e \in E(s_f), \\ m(e) \in \{L_{LE} \cup L_M \cup L_{IO}\}\}. \tag{24}$$

The performed mapping of the user circuit design elements to the basic project elements is the process of the placement of a user circuit design.

Features of the Representation of a Basic Chip Layout

The selection of specific peripheral elements and the placement of user circuit elements on a basic chip are performed based on the results of the chip layout analysis. If the prototype layout has been developed, the CAD system analyzes the layout file in the GDSII format, which contains all the necessary information, such as the real coordinates of the elements e: $m(e) \in L_{LE} \cup L_{IO} \cup L_M$, which allow the program to transfer their location on the flat representation of a basic chip $\Pi_f(\Pi) = (S_f, L_f, s_f)$, element orientations (rotations), and its geometric dimensions: width and height.

Thus, the position of each instance $m(e)$ is characterized by the anchor point coordinates in the lower left edge, orientation, and overall dimensions:

$$X_{min}(m(e)), \quad Y_{min}(m(e)), \quad O_r(m(e)), \tag{25}$$

where $O_r(m(e)) \in \{O_0, O_R, O_{XY}, O_{XYR}, O_Y, O_{XR}, O_X, O_{XR}\}$ is the orientation. At the same time, the orientation index indicates the absence (0) or the presence of rotation (R is a 90° counterclockwise rotation) and reflections relative to the X, Y axis.

A simplified layout view of a basic chip using relative element coordinates is applied to speed up software circuit prototyping. It allows us to skip the layout design stage and transfer the location of the LEs, I/O cells, and macroblocks to the CAD. Relative coordinates are generated for all the necessary elements using the set of operations developed based on the CDL netlist, schematic view, and specialized linguistic tools in the Tcl language. The generation of such coordinates is possible after changing the orientation of all element types of the basic chip to normal: $O_r(m_{ij}) = O_0$.

If at the top level of a chip prototype only logic elements of the same type are used, then the prototype can be represented in a flat view as a complete LE matrix:

$$M_f = \{m_{ij}\colon m_{ij} \in E_{LE}(s_f), \quad m(m_{ij}) \in L_{LE}, \quad i = 1,\ldots,I_f, \quad j = 1,\ldots,J_f\}, \\ E_{LE}(s_f) \subset E(s_f), \quad E_{LE}(s_f) = \{e\colon e \in E(s_f) \,\&\, m(e) \in L_{LE}\}. \tag{26}$$

If the top level of a chip prototype is represented as a matrix of LABs, then within generating coordinates of such a prototype for a more detailed representation, a simplified bilevel view $\Pi_b(\Pi) = (S_b, L_b, s_b)$ is introduced. In this view, together with the set of elements of the library level L, an intermediate level of blocks that are not included in L: $B = \{b_i\}$, $:B \cap L = \emptyset$ is allocated. Due to the fact that LABs are grouped from identical blocks, the following expression is true: $|B| = |\{b\}| = 1$. The final simplified project view $\Pi_b(\Pi) = (S_b, L_b, s_b)$ consists of the following components:

$$L_b = \{s\colon (s \in L) \wedge \varphi(s, s_m)\}; \\ S_b = \{s_b \cup L_b \cup B\}, \quad L_b \cap B = \emptyset; \\ s_b = (\mu(s_b), E(s_b), \emptyset, \emptyset, \emptyset); \\ \mu(s_b) = \mu(s_m).$$

In contrast to the representation of the prototype in CAD, when the coordinates of the prototype are generated, all the logic elements are grouped formally. In accordance with this, the set of prototype nets, as well as the set of its external pins and connections, is not parsed, but only the set of elements is used. Also, in contrast to a flat representation in CAD, in this case, not only logic elements L_{LE}, peripheral I/O elements L_{IO}, and complex-functional macroblocks L_M, but also LABs B are used:

$$S_b = \{s_b \cup L_{LE} \cup L_{IO} \cup L_M \cup B\}. \tag{27}$$

Based on this, the subset of block elements is $E_B(s_b) \subset E(s_b)$, $E_B(s_b) = \{e\colon e \in E(s_b) \,\&\, m(e) \in B\}$, and the prototype can be considered in the LAB matrix view:

$$M_b = \{m_{ij}\colon m_{ij} \in E_B(s_b), \\ m(m_{ij}) \in B, \quad i = 1,\ldots,I_b, \\ j = 1,\ldots,J_b\}. \tag{28}$$

The total number of blocks in the basic chip is determined by the block matrix size:

$$|E_B(s_b)| = |M_b| = I_b J_b. \quad (29)$$

Similarly, a block in a bilevel view consists of elements at a lower hierarchy level:

$$b = (\mu(b), E(b), \emptyset, \emptyset, \emptyset), \quad (30)$$

where $E(b) = \{e: m(e), m(e) \in L_{LE} \cup L_{Ro} \cup L_{BB}\}$. Then the subset of logic elements of the block is $E_{LE}(b) \subset E(b)$, $E_{LE}(b) = \{e: e \in E(b) \& m(e) \in L_{LE}\}$ and can be represented as an LE matrix part of the LAB:

$$M_{LE} = \{m_{ij}: m_{ij} \in E_{LE}(b), \ m(m_{ij}) \in L_{LE}, \ i = 1,\ldots,I_{LE}, \ j = 1,\ldots,J_{LE}\}. \quad (31)$$

The total number of elements in a block is determined by the matrix size M_{LE}:

$$|E_{LE}(b)| = |M_{LE}| = I_{LE} J_{LE}. \quad (32)$$

It is assumed that all logic elements in the bilevel block view are localized in blocks:

$$\forall e \in E(s_b) \cup E(b): m(e) \in L_{LE} \rightarrow e \in E(b). \quad (33)$$

In other words, there are no logic elements at the top level of the hierarchical bilevel block view:

$$\nexists e: e \in E(s_b) \ \& \ m(e) \in L_{LE}.$$

Then the total logic elements number in the basic chip is determined by the size of the matrices M_b and M_{LE}:

$$\begin{gathered} I_f = I_b I_{LE}, \\ J_f = J_b J_{LE}, \\ |E_{LE}(s_f)| = |M_f| = I_f J_f = |E_B(s_b)||E_{LE}(b)| = I_b J_b I_{LE} J_{LE}. \end{gathered} \quad (34)$$

For ease of use, we introduce the following notation for the elements of the LAB and LE sets:

$$\begin{gathered} m_{ijB} = m_{ij} \in E_B(s_b); \\ m_{ijLE} = m_{ij} \in E_{LE}(b). \end{gathered} \quad (35)$$

When the coordinates of the bilevel basic chip view are generated, it is assumed that the anchor point is the lower left corner of the chip, and the LEs are indexed from it to the upper right; i.e., $m_{\min LE}$, is the bottom left element and $m_{\max LE}$. is the top right element. Also, before generation, in addition to the known parameters, such as the number of LEs in row J_{LE} and column I_{LE} of the LAB, the following parameters are set:

—initial coordinates of the lower left chip edge corresponding to the coordinates of the lower left corner of the lowermost LE:

$$X_{\min}(s_b), \ Y_{\min}(s_b) = X_0(m_{00LE}), \ Y_0(m_{00LE}); \quad (36)$$

—the distance between LEs in the LAB:

$$\Delta X(m_{ijLE}), \ \Delta Y(m_{ijLE}); \quad (37)$$

—the LE dimensions—width and height:

$$W(m_{ijLE}), \ H(m_{ijLE}); \quad (38)$$

—the distance between LABs:

$$\Delta X(m_{ijB}), \ \Delta Y(m_{ijB}). \quad (39)$$

Based on the known parameters, the dimensions of the LAB are calculated:

$$\begin{gathered} W(m_{ijB}) = J_{LE} W(m_{ijLE}) + (J_{LE} - 1)\Delta X(m_{ijLE}), \\ H(m_{ijB}) = I_{LE} H(m_{ijLE}) + (I_{LE} - 1)\Delta Y(m_{ijLE}). \end{gathered} \quad (40)$$

Further, using the described specified and computed parameters, the coordinates are calculated for each item m_{ijB} consisting of m_{ijLE}. After each LE, the distance to the next element in the X direction and in the Y direction is taken into account. $\Delta X(m_{ijB})$ is added to the LE X coordinates after each element width $W(m_{ijB})$ and $\Delta Y(m_{ijB})$ is added to the LE Y coordinates after each element's height $H(m_{ijB})$.

It should be noted that these formulas for dimensions and coordinates are valid not only for logic elements of the matrix $M_b = \{m_{ij}: m_{ij} \in E_B(s_b), m(m_{ij}) \in B, i = 1,\ldots,I_b, j = 1,\ldots,J_b\}$, but also for the peripheral I/O elements

$$E_{IO}(s_b) \subset E(s_b), \ E_{IO}(s_b) = \{e: e \in E(s_b) \ \& \ m(e) \in L_{IO}\} \quad (41)$$

and macroblocks

$$E_M(s_b) \subset E(s_b), \ E_M(s_b) = \{e: e \in E(s_b) \ \& \ m(e) \in L_M\}. \quad (42)$$

The number of macroblocks and their location on the chip can be completely different; therefore, a structured description of generating of their coordinates will not be given. However, the peripheral I/O elements, as a rule, are located along the perimeter of the LE or LAB matrix; therefore, their initial coordinates can be described relatively to LEs. Depending on the side where the peripheral elements are located (left, right, top, bottom), the following coordinates are determined:

$$\begin{gathered}
X\left(m_{0left}\right),\ \ Y\left(m_{0left}\right),\ \ \text{where} \\
X\left(m_{0right}\right) = X\left(m_{00LE}\right) - \Delta X\left(m_{ijLE}, m_{ijIO}\right) - W\left(m_{ijIO}\right), \\
Y\left(m_{0left}\right) = Y_0\left(m_{00LE}\right) \\
X\left(m_{0right}\right),\ \ Y\left(m_{0right}\right),\ \ \text{where} \\
X\left(m_{0right}\right) = X\left(m_{00LE}\right) + J_b W\left(m_{ijB}\right) + (J_b - 1)\Delta X\left(m_{ijB}\right) + \Delta X\left(m_{ijLE}, m_{ijIO}\right) \\
Y\left(m_{0right}\right) = Y_0\left(m_{00LE}\right) \\
X\left(m_{0bottom}\right),\ \ Y\left(m_{0bottom}\right),\ \ \text{where} \\
X\left(m_{0bottom}\right) = X_0\left(m_{00LE}\right); \\
Y\left(m_{0bottom}\right) = Y\left(m_{00LE}\right) - \Delta Y\left(m_{ijLE}, m_{ijIO}\right) - H\left(m_{ijIO}\right), \\
X\left(m_{0top}\right),\ \ Y\left(m_{0top}\right),\ \ \text{where} \\
X\left(m_{0top}\right) = X\left(m_{00LE}\right) \\
Y\left(m_{0top}\right) = Y\left(m_{00LE}\right) + I_b H\left(m_{ijB}\right) + (I_b - 1)\Delta Y\left(m_{ijB}\right) + \Delta Y\left(m_{ijLE}, m_{ijIO}\right).
\end{gathered} \tag{43}$$

The other parameters for generating the coordinates of the I/O elements are identical to the LE and LAB parameters. The difference is that LE is replaced by IO, and LAB is replaced by a group of peripheral elements. The index of each PE instance is defined in accordance with the coordinate axis. The PE index on the left and right side corresponds to the Y axis and the PE index on the bottom and top side corresponds to the X axis.

At this stage, simultaneously with the generation of coordinates, the information that the element e: $m(e) \in L_{IO}$ corresponds to the external pin of the basic chip $P(s_m)$ is formed.

PRACTICAL RESULTS OF SOWTWARE CIRCUIT PROTOTYPING

Based on the formalized representation of a basic chip and a user circuit design described in subsections III.a and III.b, CAD software was developed to allow the proposed software circuit prototyping method to be applied. As an example, that demonstrates the efficiency of the presented method, the results of developing a basic chip using iterative modifications in its architecture are given. The closest analog of the original basic chip is the Altera MAX II FPGA, which has a similar structure of LE, LAB, and routing resources. The purpose of these modifications is to reduce the required amount of configuration memory and to increase the logic size of the basic circuit without downgrading the achieved routing level for the user circuit design based on an existing chip.

In the process of prototyping, the LAB structure and the routing chip architecture were modified. The routing architecture consists of the following types of interconnects: local buses, direct links (DL), R4C4 and R8C8 buses (R is the row, C is the column), long buses, and diagonal connections.

Also, in addition to changing the bit width of the presented buses, the structure of the switches and connection blocks that connect these nets to each other was modified. There are three types of such blocks in this architecture:

—switch block (SB) is a block that connects the R4C4/R8C8 buses and allows to connect direct links to these buses;

—connection block (CB) is a block in which the R4C4/R8C8 buses, direct links, and long buses into a local bus inside the LAB intersect;

—local connection block (LCB) is a block that connects signals on local buses with all the necessary LEs inside the LAB.

We will consider the functionality of all the available interconnect types in more detail. The local bus provides communication between LEs inside a LAB. It is connected to each LE separately and to the rows and columns of global interconnects. This allows direct communication between LABs and minimizes the use of global buses.

At the same time, three types of buses can be used to connect the LAB to each other within one line:

—direct communication using a local bus;

—R4 bus that connects four LABs on the left and four on the right;

Table 1. Results of software circuit prototyping of the basic chip consisting of 16 × 20 LABs

Prototype name	Previous prototype name	Description of the current prototype	Unrouted nets number, pcs.	Memory volume per LE/LAB, bit
1.0	–	Initial basic chip R4C4 = 32, R8C8 = 64, LongBus = 10, DL = 10, Local = 22	0	**291.3**/2913
1.1	1.0	**Reduction of CB capacity** R4C4 = 32, **R8C8 = 32**, LongBus = 10, DL = 10, Local = 22	114	**278.7**/2787
1.2	1.1	**Reduction of CB capacity** **R4C4 = 16**, R8C8 = 32, LongBus = 10, DL = 10, Local = 22	–	**252.5**/2525
1.3	1.1	**Reduction of CB capacity** **R4C4 = 16**, **R6C6 = 24**, LongBus = 10, DL = 10, Local = 22		**249.3**/2493
1.4	1.3	**Reduction of CB capacity** **R3C3 = 24**, **R6C6 = 24**, LongBus = 10, DL = 10, Local = 22	111	**268.5**/2685
1.5	1.4	**Reduction of CB capacity** R3C3 = 24, R6C6 = 24, LongBus = 10, DL = 10, Local = 22	113	**249.3**/2493
1.6	1.5	R3C3 = 24, R6C6 = 24, LongBus = 10, **DL = 5**, Local = 22 **Adding DL ↗ and ↙ to SB = 5** **DL (↖, ↑, ↗, ↖ to SB, ↗ to SB)** **are available at 5 upper LEs from LAB** **DL (↙, ↓, ↘, ↙ to SB, ↘ to SB)** **are available at 5 lower LEs from LAB**	114	**244.9**/2449
1.7	1.6	R3C3 = 24, R6C6 = 24, LongBus = 10, DL = 5, Local = 22 **Removing connections ↗ and ↙ to SB**	112	**233.7**/2337
1.8	1.6	R3C3 = 24, R6C6 = 24, LongBus = 10, **DL (←, →) = 10**, DL= 5, DL ↗ and ↙ (to SB) = 5, Local = 22,	113	**246.1**/2461
1.9	1.8	R3C3 = 24, R6C6 = 24, LongBus = 10, DL (←, →) = 10, DL = 5, Local = 22 **Removing connections DL ↗ and ↙ (to SB) = 5**	116	**234.9**/2349
1.10	1.6	R3C3 = 24, R6C6 = 24, LongBus = 10, DL **(↖, ↗, ↙, ↘) = 5, DL = 10**, Local = 22	108	**247.1**/2471
1.11	1.10	R3C3 = 24, R6C6 = 24, LongBus = 10, DL (↖, ↗, ↙, ↘) = 5, DL = 10, Local = 22 **Removing DL ↗ and ↙ (to SB) = 5**	114	**235.9**/2359
1.12	1.0	R4C4 = 32, R8C8 = 64, LongBus = 10, **DL = 5**, Local = 22 **Adding DL ↗ and ↙ (to SB) = 5**	112	**286.9**/2869
1.13	1.12	R4C4 = 32, R8C8 = 64, LongBus = 10, **DL = 5**, Local = 22 **Removing DL ↗ and ↙ (to SB)**	114	**274.1**/2741

Table 1. (Contd.)

Prototype name	Previous prototype name	Description of the current prototype	Unrouted nets number, pcs.	Memory volume per LE/LAB, bit
1.14	1.5	**Reduction of the connection block (CB) capacity** R3C3 = 24, R6C6 = 24, LongBus = 10, **DL = 4**, Local = 22 **Adding DL ↗ and ↙ (to SB) = 4** **DL (for directions ↖, ↑, ↗, ↖ to SB, ↗ to SB) are available for 4 upper LEs from LAB.** **DL (↙, ↓, ↘, ↙ to SB, ↘ to SB) available for 4 lower LEs from LAB** **DL (←, →) are available for 4 central LE (3, 4, 5, 6 LE)**	115	**235.3**/2353
1.15	1.14	R3C3 = 24, R6C6 = 24, LongBus = 10, DL = 4, **DL to SB = 8**, Local = 22 **DL to SB inside LAB reduced to 8**	118	**227.3**/2273
1.16	1.15	**Full switch block for DL in all directions is added** R3C3 = 24, R6C6 = 24, **LongBus = 8**, DL = 4, **DL to SB = 8**, Local = 22	112	**275.3**/2753
1.17	1.16	The LEs number in the LAB increased to 16. **R4C4 = 32, R8C8 = 64**, LongBus = 8, DL = 4, DL to SB = 8, Local = 22	0	**269**/4314
1.18	1.17	**R3C3 = 24, R6C6 = 48**, LongBus = 8, DL = 4, DL to SB = 8, Local = 22	0	**275**/4114

Table 2. Results of software circuit prototyping of the basic chip consisting of 16 × 16 LABs

Prototype name	Description of a current prototype	Memory size per LE/LAB, bit	5 maximal net lengths. Average net length	
			S38417	Ac97
2.1	The chip consisting of 16 × 16 LABs, size 16 LEs. R3C3 = 24, R6C6 = 48, LongBus = 8, DL = 4, Local = 34	**278**/4454	26/24/24/24/23 **8.56**	27/27/27/27/27 **9.97**
2.2	Added direct links that connect LongBus and IO blocks	**278**/4454	27/27/27/23/22 **8.52**	37/37/37/37/37 **10.25**
2.3	Switch block (SB) reduction	**264**/4230	19/19/19/19/19 **7.54**	25/25/25/25/25 **8.42**
2.4	In the switch block (SB), the turns of the R3C3 and R6C6 buses have been reduced from 8 bits to 4	**259**/4150	22/19/19/19/19 **7.55**	28/28/27/27/27 **8.66**
2.5	Local connections within LAB reduced from 10 bits to 8	**252**/4042	22/19/19/19/19 **8.31**	31/28/28/27/27 **9.60**
2.6	Local connection block capacity is reduced < 75%	**244**/3914	22/22/19/19/19 **8.31**	28/28/28/28/27 **9.68**

—R8 bus that connects eight LABs on the left and four on the right.

Direct communication gives access to local buses of neighboring LABs, which are located on the left and right, and also provides a fast data transfer between LABs and/or I/O units without connecting to the R4 and R8 buses. Each LAB has connections to the R4/R8 buses both to the left and to the right.

The interconnects structure in columns and rows is similar to each other. The only difference is that instead of R4/R8 buses, C4/C8 buses are used. This connection structure allows connecting neighboring

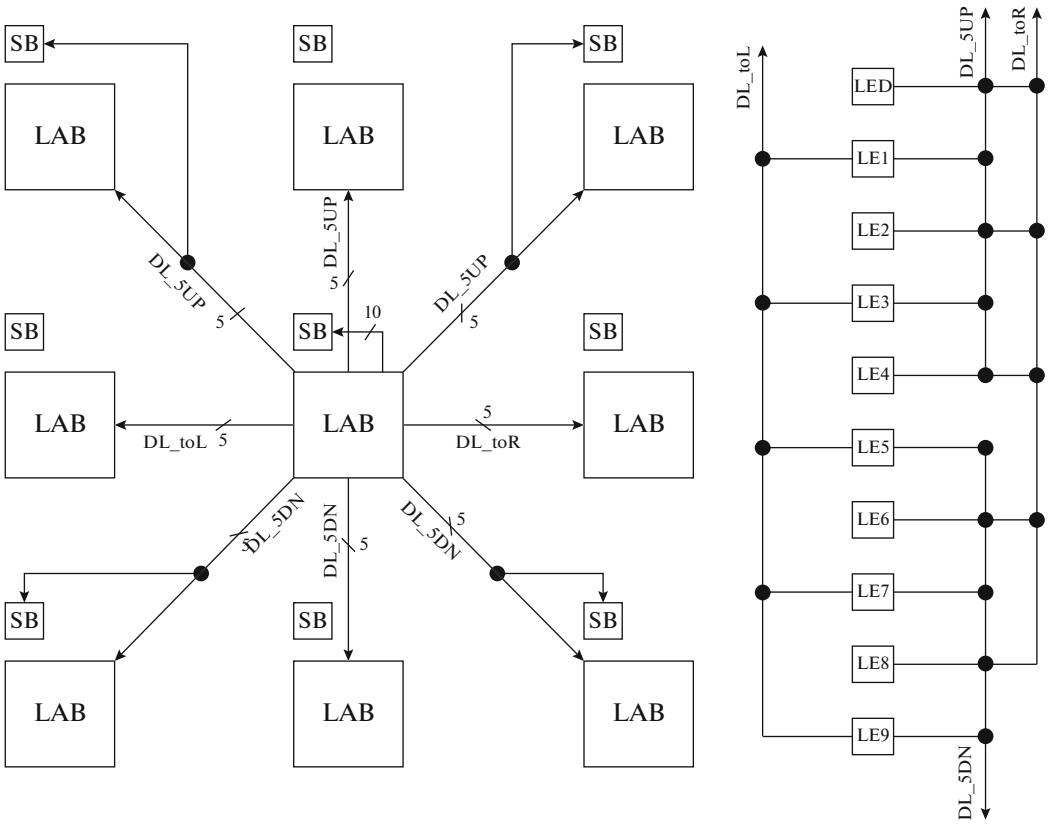

Fig. 2. Schematic representation of the direct links structure in prototype 1.6.

LABs (four/eight neighboring LABs upward and the same number of LABs downward) within one column.

Also, the regular connection structure in the form of rows and columns with a fixed length allows to predict the propagation delay time accurately.

Long buses contained in the architecture cross the entire column or row of the chip and connect the distant LEs.

A prototype with the following routing architecture characteristics was taken as the initial circuit: the width of the R4/C4 buses is 32 bits, the width of the R8/C8 buses is 64 bits, the length of long buses is 10 bits, direct links are 10 bits, and the local bus is 22 bits. In this case, the local connection block capacity is not full (not 100%). Local connection block structure is sparse and the capacity is reduced to 75%. This basic circuit consists of 16 columns and 20 lines of LAB. At the same time each LAB consists of 10 LE. The total area of the circuit is 3200 LE.

Prototyping was performed using the s38417 test user circuit design from the ISCAS'89 set [26]. The result of the logic synthesis is 3184 LEs and 3215 wires.

The prototyping results are shown in Table 1. It consists of columns with the names of the current prototype and previous prototype, a description of the current prototype, and the analysis results of the circuit implementation in it. The results are presented in the form of unrouted nets number and the configuration memory volume obtained per LAB and, on average, per element of this block.

Table 1 shows that in prototypes 1.1–1.5, memory reduction was achieved by reducing the connection block and reducing the length and width of the R/C buses. In prototypes 1.2–1.3, the used length and width of R/C buses were found to be unacceptable. With these bus parameters, the routability drops to almost 0 for any user circuit design of any size.

In prototypes 1.6–1.15, an attempt was made to reduce the amount of configuration memory by reducing direct links without degrading routability. First, the direct link width was reduced to 5 bits. Second, only one half of the LEs has direct links upward and upward along the diagonals; and the other half, only downward links and downward along the diagonals. The available connections of prototype 1.6 are shown in more detail in Fig. 2.

In prototype 1.16 a full switch block is again added, because further reduction of direct links will only worsen routability. Such a block allows each LE in the LAB to connect to the nearest LE. At the same time, the long buses' width has been reduced to 8 bits in order not to increase the amount of configuration memory.

The routability is reduced by ~113 unrouted nets on prototypes 1.1–1.16 (in contrast to the original chip) but is improved by increasing the length and width of the R/C buses, as well as by increasing the LAB size to 16 LEs.

When full routability is achieved, a reduction in the maximum and average net length is added to the prototyping goals in addition to reducing the amount of memory. The next stages of software circuit prototyping are shown in Table 2, taking the new goals set into consideration. Here, prototype 1.16 is taken as the initial basic chip with an increased LAB size of up to 16 LEs, an increased number of LABs, and a proportional increase in the width of the routing buses. The total area of the circuit is 4096 LEs. The table does not contain a column with the previous prototype name, since modifications are made sequentially to the previous prototype. Also, the table does not contain a column with the number of unrouted nets, since all test circuits are completely routed in all prototypes.

The increase of the prototype size allows to test larger user design. In this case, ac97 was used for testing [27]. The size of ac97 after logic synthesis was 3732 LEs and 3821 wires.

Table 2 also demonstrates that at this stage of prototyping, memory is reduced due to the reduction of SBs and LCBs, as well as a small change in the width of local connections and the width of the R3C3 and R6C6 buses at the intersection of a row and a column.

The result of the software prototyping is the basic chip developed from prototype 2.6. The average memory size per LE in this prototype is 47.3 bits less than the parent prototype. At the same time, the initial level of interconnection routability is not lost and the total size of the chip is increased by 896 LEs.

Thus, software circuit prototyping made it possible to evaluate the architecture of the basic chip before its layout design and to obtain an FPGA that meets all the specified requirements.

CONCLUSIONS

This paper presents a new stage in the design flow for reconfigurable and heterogeneous SoCs and FPGAs called software circuit prototyping. This stage allows to evaluate the basic chip architecture before developing the chip layout. At the same time, the paper describes the method developed for software circuit prototyping and a formalized representation of the RSoC and FPGA circuitry in CAD tools, which provides flexible and prompt software configuration for a loaded circuit.

The stage of loading a basic chip is also considered in detail and the features of the analysis and processing of the RSoC and FPGA layout in CAD are presented. The practical results of using the developed method of software FPGA architecture prototyping are demonstrated. Possible architecture parameters and characteristics, based on which the obtained prototypes can be compared, are described.

REFERENCES

1. Krasnikov, G.Ya., Enns, V.I., et al., Development and manufacture of a library of analog IP blocks for use as part of very large-scale integrated circuits “System on a Chip” at a domestic enterprise using technology with minimum topological norms of no more than 0.18 microns, *Research Report,* Moscow: Zelenograd, 2017.
2. Enns, V.I., SoC, BMK or FPGA: Choice of digital integrated circuit option, *Kompon. Tekhnol.,* 2018, no. 4, pp. 100–102.
3. Krasnikov, G.Ya., The capabilities of microelectronic processes with 5 nm critical dimension and less, *Nanoindustriya,* 2020, vol. 13, no. S5-1 (102), pp. 13–19.
4. Krasnikov, G.Ya., Panasenko, P.V., Volosov, V.A., and Shcherbakov, N.A., Trends in the development of technology of complex functional heterointegrated ECB, in *Tr. Mezhdunarodn. foruma 'Mikroelektronika-2018', 4-ya Mezhdunarodnaya nauchnaya konferentsiya 'Elektronnaya komponentnaya baza i mikro elektronnye moduli'* (Proceedings of the International Forum Microelectronics-2018, 4th International Conference on Electronic Component Base and Microelectronic Modules), Alushta: Tekhnosfera, 2018, pp. 341–344.
5. Li, X., Yang, H., and Zhong, H., Use of VPR in design of FPGA architecture, in *Proceedings of the 8th International Conference on Solid-State and Integrated Circuit Technology, 2006,* Shanghai, China: IEEE, 2006, pp. 1880–1882.
6. Luu, J. et al., VPR 5.0: FPGA CAD and architecture exploration tools with single-driver routing, heterogeneity and process scaling, in *ACM Transactions on Reconfigurable Technology and Systems (TRETS),* Monterey, CA: ACM, 2008, pp. 133–142.
7. Parvez, H. et al., A new coarse-grained FPGA architecture exploration environment, in *Proceedings of the 2008 International Conference on Field-Programmable Technology,* Taipei, Taiwan: IEEE, 2008, pp. 285–288.
8. Kannan, P., Balachandran, S., and Bhatia, D., On metrics for comparing routability estimation methods for FPGAs, in *Proceedings of the 2002 Design Automation Conference,* New Orleans, LA, USA: IEEE, 2002, pp. 70–75.
9. Gao, Hai-Xia et al., A novel Monte-Carlo method for FPGA architecture research, in *Proceedings of the 7th International Conference on Solid-State and Integrated Circuits Technology, 2004,* Bejing, China: IEEE, 2004, pp. 1944–1947.
10. Doman, D., *Engineering the CMOS Library: Enhancing Digital Design Kits for Competitive Silicon,* New York: Wiley, 2012.
11. Amos, D., Lesea, A., and Richter, R., *FPGA-Based Prototyping Methodology Manual: Best Practices in Design-for-Prototyping,* USA: Synopsys Press, 2011.
12. Ohba, N. and Takano, K., An SoC design methodology using FPGAs and embedded microprocessors, in *Proceedings of the 41st Annual Design Automation Conference DAC'04,* New York: Assoc. Comput. Machinery, 2004, pp. 747–752.

13. Chochaev, R.Zh., Zheleznikov, D.A., Ivanova, G.A., Gavrilov, S.V., and Enns, V.I., FPGA routing architecture estimation models and methods, *Izv. Vyssh. Uchebn. Zaved., Elektron.,* 2020. vol. 25, no. 5, pp. 410–422.
14. Gandhare, S. and Karthikeyan, B., Survey on FPGA architecture and recent applications, in *Proceedings of the 2019 International Conference on Vision Towards Emerging Trends in Communication and Networking ViTECoN,* Vellore, India, 2019, pp. 1–4.
15. MAX II Device Handbook, Altera Corp. www.intel.com/content/dam/www/programmable/us/en/pdfs/literature/hb/max2/max2_mii5v1.pdf. Accessed April 1, 2021.
16. UltraScale Architecture Configurable Logic Block User Guide, Xilinx Inc. www.xilinx.com/support/documentation/user_guides/ug574-ultrascale-clb.pdf. Accessed April 01, 2021.
17. ProASIC3 nano FPGA Fabric User's Guide, Microsemi Corp., http://www.ibselectronics.com/ibsstore/datasheet/Microsemi/PA3_nano_UG.pdf. Accessed April 01, 2021.
18. GDSII™ Stream Format Manual, Release 6.0, Calma Comp., http://bitsavers.informatik.uni-stuttgart.de/pdf/calma/GDS_II_Stream_Format_Manual_6.0_Feb87.pdf. Accessed April 01, 2021.
19. Gavrilov, S.V., Zheleznikov, D.A., Zapletina, M.A., Khvatov, V.M., Chochaev, R.Zh., and Enns, V.I., Layout synthesis design flow for special-purpose reconfigurable systems-on-a-chip, *Russ. Microelectron.,* 2019, vol. 48, no. 3, pp. 176–186.
20. Vasil'ev, N.O., Tiunov, I.V., and Ryzhova, D.I., Method of logical resynthesis of circuits in the design route on FPGA, *Probl. Razrab. Persp. Mikro- Hanoelektron. Sist.,* 2020, no. 4, pp. 39–44.
21. Ivanova, G.A., Ryzhova, D.I., Gavrilov, S.V., Vasil'ev, N.O., and Stempkovskii, A.L., Methods and algorithms for the logical-topological design of microelectronic circuits at the valve and inter-valve levels for promising technologies with a vertical transistor gate, *Russ. Microelectron.,* 2019, vol. 48, no. 3, pp. 167–175.
22. Gavrilov, S.V., Zheleznikov, D.A., Chochaev, R.Zh., and Khvatov, V.M., Partitioning algorithm based on simulated annealing for reconfigurable systems-on-chip, *Probl. Razrab. Persp. Mikro- Nanoelektron. Sist.,* 2018, no. 1, pp. 199–204.
23. Gavrilov, S.V., Zheleznikov, D.A., Chochaev, R.Zh., and Enns, V.I., The modification of simulated annealing-based placement algorithm for reconfigurable systems-on-chip, *Elektron. Tekh., Ser. 3: Mikroelektron.,* 2018, no. 4 (172), pp. 55–61.
24. Zapletina, M.A., Zheleznikov, D.A., and Gavrilov, S.V., The hierarchical approach to island style reconfigurable system-on-chip routing, *Probl. Razrab. Perspekt. Mikro- Nanoelektron. Sist.,* 2020, no. 3, pp. 16–21.
25. Francis, R.J., A tutorial on logic synthesis for lookup-table based FPGAs, in *Proceedings of the 1992 IEEE/ACM International Conference on Computer-Aided Design ICCAD'92,* Washington, DC, USA: IEEE Comput. Soc., 1992, pp. 40–47.
26. Brglez, F., Bryan, D., and Kozminski, K., Combinational profiles of sequential benchmark circuits, in *Proceedings of the IEEE International Symposium on Circuits and Systems, Portland, OR, USA, 1989,* vol. 3, pp. 1929–1934.
27. Usselmann, R., AC 97 Controller IP Core. https://opencores.org/projects/ac97. Accessed April 3, 2021.