KHARKOV NATIONAL UNIVERSITY OF RADIOELECTRONICS

Proceedings of IEEE East-West Design & Test Symposium (EWDTS'2012)

Technically Co-Sponsored by

Kharkov, Ukraine, September 14 – 17, 2012

IEEE EAST-WEST DESIGN AND TEST SYMPOSIUM 2012 COMMITTEE

General Chairs
V. Hahanov – Ukraine
Y. Zorian – USA

General Vice-Chairs
R. Ubar - Estonia
P. Prinetto - Italy

Program Chairs
S. Shoukourian – Armenia
D. Speranskiy – Russia

Program Vice-Chairs
Z. Navabi – Iran
M. Renovell – France

Publicity Chair's
G. Markosyan - Armenia
S. Mosin - Russia

Public Relation Chair
V. Djigan - Russia

Program Committee
E. J. Aas - Norway
J. Abraham - USA
M. Adamski - Poland
A.E.Mohamed Mohamed - Egypt
A . Barkalov - Poland
R. Bazylevych - Ukraine
A. Chaterjee - USA
V. Djigan - Russia
A. Drozd - Ukraine
E. Evdokimov - Ukraine
E. Gramatova - Slovakia
A. Ivanov - Canada
M. Karavay - Russia
V. Kharchenko - Ukraine
K. Kuchukjan - Armenia
W. Kuzmicz - Poland
A. Matrosova - Russia
V. Melikyan - Armenia
L. Miklea - Romania
O. Novak - Czech Republic
Z. Peng - Sweden
A. Petrenko - Ukraine
J. Raik - Estonia
A. Romankevich - Ukraine
A. Ryjov - Russia
R. Seinauskas - Lithuania
S. Sharshunov - Russia
A. Singh - USA
J. Skobtsov - Ukraine
V. Tverdokhlebov - Russia
V. Vardanian - Armenia
V. Yarmolik - Byelorussia

Steering Committee
M. Bondarenko - Ukraine
V. Hahanov - Ukraine
R. Ubar - Estonia
Y. Zorian - USA

Organizing Committee
S. Chumachenko - Ukraine
E. Litvinova – Ukraine
V. Kharchenko – Ukraine

EWDTS 2012 CONTACT INFORMATION

Prof. Vladimir Hahanov
Design Automation Department
Kharkov National University of Radio Electronics,
14 Lenin ave,
Kharkov, 61166, Ukraine.

Tel.: +380 (57)-702-13-26
E-mail: hahanov@kture.kharkov.ua
Web: www.ewdtest.com/conf/

10th IEEE EAST-WEST DESIGN & TEST SYMPOSIUM (EWDTS 2012)

Kharkov, Ukraine, September 14-17, 2012

The main target of the **IEEE East-West Design & Test Symposium** (EWDTS) is to exchange experiences between scientists and technologies of Eastern and Western Europe, as well as North America and other parts of the world, in the field of design, design automation and test of electronic circuits and systems. The symposium is typically held in countries around the Black Sea, the Baltic Sea and Central Asia region. We cordially invite you to participate and submit your contributions to EWDTS'12 which covers (but is not limited to) the following topics:

- Analog, Mixed-Signal and RF Test
- Analysis and Optimization
- ATPG and High-Level Test
- Built-In Self Test
- Debug and Diagnosis
- Defect/Fault Tolerance and Reliability
- Design for Testability
- Design Verification and Validation
- EDA Tools for Design and Test
- Embedded Software Performance
- Failure Analysis, Defect and Fault
- FPGA Test
- HDL in test and test languages
- High-level Synthesis
- High-Performance Networks and Systems on a Chip
- Low-power Design
- Memory and Processor Test
- Modeling & Fault Simulation
- Network-on-Chip Design & Test
- Modeling and Synthesis of Embedded Systems
- Object-Oriented System Specification and Design
- On-Line Testing
- Power Issues in Design & Test
- Real Time Embedded Systems
- Reliability of Digital Systems
- Scan-Based Techniques
- Self-Repair and Reconfigurable Architectures
- Signal and Information Processing in Radio and Communication Engineering
- System Level Modeling, Simulation & Test Generation
- System-in-Package and 3D Design & Test
- Using UML for Embedded System Specification
- CAD and EDA Tools, Methods and Algorithms
- Design and Process Engineering
- Logic, Schematic and System Synthesis
- Place and Route
- Thermal, Timing and Electrostatic Analysis of SoCs and Systems on Board
- Wireless and RFID Systems Synthesis
- Digital Satellite Television

The Symposium will take place in Kharkov, Ukraine, one of the biggest scientific and industrial center. Venue of EWDTS 2012 is Kharkov National University of Radioelectronics was founded 81 years ago. It was one of the best University of Soviet Union during 60th - 90th in the field of Radioelectronics. Today University is the leader among technical universities in Ukraine.

The symposium is organized by Kharkov National University of Radio Electronics and Science Academy of Applied Radio Electronics http://anpre.org.ua/ in cooperation with Tallinn University of Technology. It is technically co-sponsored by the IEEE Computer Society Test Technology Technical Council (TTTC) and financially supported by Trades Committee of Kharkov National University of Radioelectronics and Trades Committee of Students, Aldec, Synopsys, Kaspersky Lab, DataArt Lab, Tallinn Technical University.

CONTENTS

Synthesis of Qubit Models for Logic Circuits

Wajeb Gharibi [1], Zaychenko S.A.[2], Dahiri Farid [2], Hahanova Yu.V.[2], Guz O.A.[3], Ngene Christopher Umerah (Nigeria) [2], Adiele Stanley (Nigeria) [2]

[1]Jazan University, Kingdom of Saudi Arabia, gharibiw2002@yahoo.com
[2]Computer Engineering Faculty, Kharkov National University of Radioelectronics, Kharkov, Ukraine, hahanov@kture.kharkov.ua
[3]Road Transport Faculty, Donetsk Institute of Road Transport, Donetsk, Ukraine, olesya_guz@ukr.net

Abstract

Qubit (quantum) structures of data and computational processes for significantly improving performance when solving problems of discrete optimization and fault-tolerant design are proposed. We describe superpositional method for synthesizing cube of functionality for its implementation in the structural components of programmable logic chips. The estimates of synthesis time, as well as hardware costs for creating qubit models of logic circuits are represented.

1. Introduction

Quantum computing becomes interesting for cyberspace analysis, creating new Internet technologies and services, which is explained by their alternative to the existing models of computing processes. Market appeal of quantum (qubit) models is based on the high parallelism when solving almost all discrete optimization problems, factoring, minimization of Boolean functions, effective compression of data, their compact representation and teleportation, fault-tolerant design [1-9] through significant increase in hardware cost. But now it is acceptable, because there are problems of use silicon chip, which contains up to 1 billion gates on a substrate thickness 5 microns. At that modern technologies allow creating a package (sandwich) containing up to 7 chips, which is comparable with the quantity of the human brain neurons. Practically, through-silicon via (TSV) connection is based on the technological capability of drilling about 10 thousand through vias in 1 square centimeter of wafer or die. Layout the indicated volume of useful functionality on chip is currently problematic. So, it is necessary to develop hardware-focused models and methods for creating high-speed tools of parallel solving real world problems. Considering the discreteness and multiple-valuedness of the alphabets for description of information processes, the parallelism, inherent in the quantum computing, is particularly actual when developing effective and intelligent engines for cyberspace or Internet [5], tools for synthesis of fault-tolerant digital primitives and systems [6], testing and simulation of digital systems-on-chips [7-9], technologies for information and computer security [3-4], brain-like models for computing, analysis and synthesis of linguistic constructions [7-8].

2. Qubit, quantum models of data and computing processes

A quantum computer is designed for fault-tolerant design and solving optimization problems by way of the brute-force method through the use of set theory. A set of elements in the traditional computer is orderly, because each byte has its own address. Therefore, the set-theoretical operations are reduced to the enumeration of all the addresses of primitive elements. Address order of data structures useful for applications where model components can be strictly ranked, which makes it possible to carry out their analysis in a single pass (a single iteration). If there is not order in the structure, for example, the set of all subsets, the classical model of memory and computational processes disimprove the analysis time of primitive association equal by the rank, or processing of associative groups is ineffective. What can be offered for unordered data instead of the strict order? Processor, where the unit cell is the image or pattern of the universe of n primitives, which generates $Q = 2^n$ all possible states of a cell as a power set or the set of all subsets. Direct solution about creating such cell is based on unitary positional coding states of primitives

that form the set of all subsets and in the limit the universe of primitives by superposition of last ones. For instance, four primitives create boolean, containing sixteen states (combinations), with four bits:

$B^2(Y)$ = {Q=(1000), E=(0100), H=(0010), J=(0001), O={Q,H}=(1010), I={E,J}=(0101), A={Q,E}=(1100), B={H,J}=(0011), S={Q,J}=(1001, P={E,H}=(0110), C={E,H,J}=(1110), F={Q,H,J}=(1011), L={Q,E,J}=(1101), V={Q,E,H}=(1110), Y={Q,E,H,J}=(1111), U=(0000)}.

Operations on the symbols of set-theoretic alphabet can be reduced to logical commands and, or, not, xor, which form the functionally complete basis, according to Post's theorem.

Another interpretation of the power set of four primitives (binary codes: 00, 01, 10, 11) creates 16 different functions of two variables. At the same time last table can be represented as symbol codes of multi-valued alphabet, which are easy to operate for solving problems of Boolean functions synthesis and analysis:

Q	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
E	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
H	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
J	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
	∅	J	H	B	E	I	P	C	Q	S	O	F	A	L	V	Y

This table can easily be constructed for any number of primitives (n=2, 3,4, 5, 6, 7, 8 …), where the set-theoretic operations on symbols can be reduced to logical operations on vectors. The multiple-valuedness of alphabet symbols has a positive effect on the minimization of Boolean functions. For example, a compact representation of input variables states for individual Boolean functions of two variables in their coding by symbols of 16-digit alphabet has at most two cubes of multivalued coverage.

$$\begin{array}{|c|c|}\hline 00&0\\01&0\\10&0\\11&0\\\hline\end{array}=\begin{array}{|c|c|}\hline Q&0\\E&0\\H&0\\J&0\\\hline\end{array}=\begin{array}{|c|c|}\hline Y&0\\\hline\end{array}=\begin{array}{|c|c|}\hline 1111&0\\\hline\end{array};$$

$$\begin{array}{|c|c|}\hline 00&0\\01&0\\10&0\\11&1\\\hline\end{array}=\begin{array}{|c|c|}\hline Q&0\\E&0\\H&0\\J&1\\\hline\end{array}=\begin{array}{|c|c|}\hline V&0\\J&1\\\hline\end{array}=\begin{array}{|c|c|}\hline 1110&0\\0001&1\\\hline\end{array};$$

$$\begin{array}{|c|c|}\hline 00&0\\01&0\\10&1\\11&0\\\hline\end{array}=\begin{array}{|c|c|}\hline Q&0\\E&0\\H&1\\J&0\\\hline\end{array}=\begin{array}{|c|c|}\hline F&0\\H&1\\\hline\end{array}=\begin{array}{|c|c|}\hline 1101&0\\0010&1\\\hline\end{array};$$

$$\begin{array}{|c|c|}\hline 00&0\\01&0\\10&1\\11&1\\\hline\end{array}=\begin{array}{|c|c|}\hline Q&0\\E&0\\H&1\\J&1\\\hline\end{array}=\begin{array}{|c|c|}\hline A&0\\B&1\\\hline\end{array}=\begin{array}{|c|c|}\hline 1100&0\\0011&1\\\hline\end{array};$$

$$\begin{array}{|c|c|}\hline 00&1\\01&1\\10&1\\11&1\\\hline\end{array}=\begin{array}{|c|c|}\hline Q&1\\E&1\\H&1\\J&1\\\hline\end{array}=\begin{array}{|c|c|}\hline Y&1\\\hline\end{array}=\begin{array}{|c|c|}\hline 1111&1\\\hline\end{array};$$

$$\begin{array}{|c|c|}\hline 00&0\\01&0\\10&1\\11&0\\\hline\end{array}=\begin{array}{|c|c|}\hline Q&0\\E&0\\H&1\\J&0\\\hline\end{array}=\begin{array}{|c|c|}\hline F&0\\H&1\\\hline\end{array}=\begin{array}{|c|c|}\hline 1101&0\\0010&1\\\hline\end{array};$$

$$\begin{array}{|c|c|}\hline 00&0\\01&1\\10&1\\11&1\\\hline\end{array}=\begin{array}{|c|c|}\hline Q&0\\E&1\\H&1\\J&1\\\hline\end{array}=\begin{array}{|c|c|}\hline Q&0\\C&1\\\hline\end{array}=\begin{array}{|c|c|}\hline 1000&0\\0111&1\\\hline\end{array};$$

$$\begin{array}{|c|c|}\hline 00&0\\01&1\\10&1\\11&0\\\hline\end{array}=\begin{array}{|c|c|}\hline Q&0\\E&1\\H&1\\J&0\\\hline\end{array}=\begin{array}{|c|c|}\hline S&0\\P&1\\\hline\end{array}=\begin{array}{|c|c|}\hline 1001&0\\0110&1\\\hline\end{array}.$$

Thus, the transition from binary vectors of the input signals to the symbols of the closed multivalued alphabet provides a fundamentally new opportunity to minimize the cubic coverages (truth tables), which will always be no more than two cubes $f(X)=\{C_1,C_0\}$, $C_1 \cup C_0 = U$, $C_1 = \overline{C}_0$, forming the unit and zero output functions by two complementary symbols, together forming the symbol universe. The power of the primitive universe $\text{card}U = n$ forms the total number of states $Q = 2^n$ or symbols derived from them, which are encoded by 2^n bits. Subsequent binary encoding of input symbol for each of two cubes allows implementing a functional primitive as close as possible to memory element of programmable logic devices (PLD), where the input word of logical element is the address of the memory cell (bit), in which the output state is recorded. However, the truth table in the form of memory is DNF, which is irreversible for the solving the problem of inverse implication. In this case, way out could be in explicitly definition of functionality in the form of a cubic coverage, and precisely two cubes of coverage specifying all the possible solutions by the inputs. Moreover, all logic elements are one input, where the input is register variable or n-bit vector that generates an address of memory storing $Q = 2^n$ bits as the values of function $Y = f(A) = f(x_1, x_2, ..., x_i, ..., x_n)$.

A qubit is a binary vector containing n bits, to define the power set (the set of all subsets) of the states $Q = 2^n$ based on the use of n primitive symbols (elements). Qubit is a set of n equivalent bits forming n primitives by unit value to describe $Q = 2^n$ states,

which constitute the power set – the set of all subsets of n primitives.

There are no numbers! All bits of qubit are equal when creating primitives. Any set-theoretic operation is executed in one cycle, which is impossible if the association of primitives is defined in countable (ordered) space of computer memory. Metric (vector and scalar) for analyzing distances proposed in [11,13] is absolutely suitable for determining the interaction of multi-valued (binary) qubit objects, processes and phenomena, by using xor-operation.

Ideally, the use of qubit structure allows representing any functionality in the form of two cubes that are bound to zero and one. These cubes form CNF and DNF respectively. The problem can be simplified further by excluding zero and one from consideration, having them implicitly in mind. At that, two cubes forming the input conditions will always be mutually inverse because they complement each other to the universe of primitives. Therefore, it is necessary to leave only one symbol, and hence one binary code, which is the truth table of (two-input) functional primitive:

$$\begin{array}{|cc|}\hline 00&0\\ 01&1\\ 10&1\\ 11&0\\ \hline\end{array}=\begin{array}{|cc|}\hline Q&0\\ E&1\\ H&1\\ J&0\\ \hline\end{array}=\begin{array}{|cc|}\hline S&0\\ P&1\\ \hline\end{array}=\begin{array}{|cc|}\hline 1001&0\\ 0110&1\\ \hline\end{array}\rightarrow P=\boxed{0110}$$

$$Y = P = E \vee H = A_1 \vee A_2 = \overline{x}_1 x_2 \vee x_1 \overline{x}_2.$$

The resulting vector is interpreted here not only as a set of addressable bits, but as a cube, forming unit value of the primitive output on which parallel vector logical operations can be performed. Otherwise, a set of input vectors can be transformed to a single vector, where each position (address) corresponds to the input combination and the value of position – to the state of function output. This makes it possible to reduce the exponential (2^n) computational complexity of cubic coverage synthesis for functionality, based on the specified truth tables of the structural components of a digital device, to a linear function by increasing the number of bits when setting the variables from n up to 2^n bits.

Thus, on the grounds of the introduced qubit data structures and Hasse-model of the computing we can make some conclusions:

1) Quantum Computer was created the experts in the field of quantum mechanics, who introduced the idea of creating a non-numeric computer, based on analog representation of information.

2) The introduced notion of a qubit corresponds to the Boolean primitives, which is the ideal nonnumeric form of object component description for analysis, synthesis and optimization of discrete objects.

3) The forms of qubit representation are the following: 1. The symbols of the universe of primitives, which generate the set of all subsets (power set). 2. Binary vectors, where the power set is a combination of unit values. 3. Hasse diagram, which forms the power set of all possible solutions on the graph. 4. Full transition graph, which determines the set of all subsets of transitions in the form of arcs. 5. The geometric representation in a plane for a qubit in the form of points and segments corresponding to the Boolean.

4) In practice, more than 90% of all IT-industry problems associated with information retrieval in cyberspace, pattern recognition and decision-making relates to the field of discrete mathematics, where it is difficult to find a place of numerical arithmetic.

5) It is necessary to create brain-like parallel associative logic (quantum) processors, which effectively use Boolean (qubit) primitives or elements (sets) to solve problems of discrete mathematics.

6) Set-theoretic operations have to be replaced the isomorphic logical instructions (and, or, not, xor) for the subsequent creating a new system of parallel qubit programming to solve logic and optimization problems.

7) Another solution for organization computing is associated with topological representation of the qubit, where the elements are the geometric shapes.

8) Nonnumeric problems, focused to the proposed processor are the following: minimization of forms of Boolean functions, when describing complex systems; searching paths in the graph; testing and diagnosis of digital systems; combinatorial studies of processes and phenomena; intelligent data searching, pattern recognition and decision making; discretization of fuzzy models and methods, when creating the intelligence.

3. Synthesis of the functionality cube by the method of superposition

Formally, there are two ways for synthesis of the functionality of the digital circuit with n inputs. The first one creates a vector of the length n, by filling in zero and unit values of the function on the basis of the direct simulation of all $q = m \times 2^n$ input signals on m primitives. The second one is the superposition method for cubes of primitive elements. It also provides determining of all coordinates of the output state vector by superposition of m cubic coverages comprising the

circuit structure. In this case, the computational complexity of obtaining the functional coverage is equal to $q = 2 \times m$ provided that the structural components of the circuit had previously been ranked in the order of signal propagation and coverage of each primitive has 2 cubes.

An instance of getting coverage by the second way for the functionality $f(X) = (X_1X_2) \vee (X_3X_4)$ that involves three two-input primitives (and, or, or) is presented below. For this purpose, two coverages: C (and) = 0001, C (or) = 0111 are used. Their interaction as the Cartesian product concerning the binary operation «or» generates the following output:

$$f(X) = (X_1X_2) \vee (X_3X_4) = (0001) \vee (0111) =$$

$\vee$	0	1	1	1
0	0	1	1	1
= 0	0	1	1	1 = (0111 0111 0111 1111).
0	0	1	1	1
1	1	1	1	1

Forming the coverage for function, where there are redundant or irrelevant variables, is of interest:

$$f(X) = (X_1X_2) \vee X_2 = \begin{bmatrix} X_1 & & X_2 \\ 0 & \wedge_\times & 0 \\ 1 & & 1 \end{bmatrix} \vee_\times \begin{matrix} X_2 \\ 0 \\ 1 \end{matrix} = \begin{bmatrix} X_1 \wedge X_2 \\ 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} \begin{matrix} X_2 \\ 0 \\ \vee\ 1 \\ 0 \\ 1 \end{matrix} =$$

$$= \begin{matrix} (X_1 \wedge X_2)X_2 \\ 0 \\ 1 \\ 0 \\ 1 \end{matrix} \begin{matrix} f(X_1,X_2) \\ 0 \\ 1 \\ 0 \\ 1 \end{matrix} = \bar{X}_1X_2 \vee X_1X_2 = \begin{matrix} f(X_1,X_2) = X_2 \\ 0 \\ 1 \end{matrix}.$$

Here, the procedure for obtaining the cube of output values is added to the minimization step, which can greatly simplify the logical structure by eliminating the irrelevant variables.

The process model for obtaining the coverage of the logical functionality through its consistent expansion of n variables consists of the following items: 1) Performing the Cartesian product $\{\vee_\times, \wedge_\times, \oplus_\times\}$ for logic function of two (n) variables to form vector of output values of the length $p = 2^n$. Here every bit of the vector of one variable X_i interacts by logic operation with every bit of the vector of other variable X_j:

$$\begin{bmatrix} X_1 & & X_2 \\ 0 & \wedge_\times & 0 \\ 1 & & 1 \end{bmatrix} = \begin{matrix} X_1 \wedge X_2 \\ 0 \\ 0 \\ 0 \\ 1 \end{matrix}$$

The dimension of the resulting vector is $p = p_i \times p_j$, where p_i, p_j are width of binary vectors for the corresponding variables. 2) Consistent implementation of Cartesian (vector) logic operations on all primitives (logic variables): $P = \prod_{i=1}^{n} P_i$ for obtaining the cube of logical functionality of the maximum dimension $p = 2^n$. 3) Minimizing the length of functionality cube by excluding irrelevant variables; elimination of m variables reduces in 2^m times the original dimension of the cube (vector) of circuit operation. 4) Exclusion of conflicting input patterns, if the terms of the logical function have the same variables.

To reduce the computational operations it is advisable to have a library of output functions for all logic operations, which occur in the functionality. In fact, these types for each circuit are no more than 10. For example, for two-output circuit shown in Fig. 1, there is only one type – and-not.

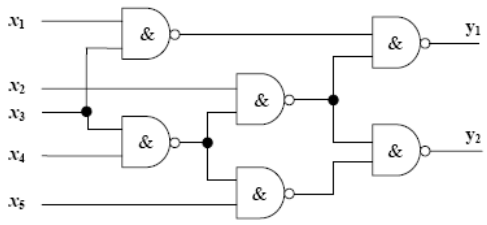

Fig. 1. Circuit structure from ISCAS library

Implementation of a process model for synthesizing the general functionality will be presented the following items: 1. Create a library of primitives in the form of cubes of their functionalities: $C_i = (0001), i = \overline{1,6}$. For this circuit there is one type of two-input primitive:

$$C_{\text{and-not}} = \begin{bmatrix} \overline{X_1X_2} & Y \\ 00 & 1 \\ 01 & 1 \\ 10 & 1 \\ 11 & 0 \end{bmatrix} \begin{matrix} C \\ 1 \\ = 1 = (1110). \\ 1 \\ 0 \end{matrix}$$

2. The procedure for the synthesis of the functionality cube for the circuit structure by superposition of the cubes of primitive elements is as follows:

$$C_5(Y_1) = C_1\overline{(X_2C_2)}^{C_3} = \overline{\overline{(X_1X_3)}\,\overline{[X_2\overline{(X_3X_4)}]}};$$

$$C_6(Y_2) = \overline{X_2C_2}^{C_3}\overline{(C_2X_5)}^{C_4} = \overline{X_2\overline{X_3X_4}\,\overline{X_3X_4}X_5}$$

$$C_5 = \begin{matrix} \overline{X_1X_3} \\ 1 \\ 1 \\ 1 \\ 0 \end{matrix} \overline{\wedge}_{\times} \begin{bmatrix} & \overline{X_3X_4} \\ X_2 & 1 \\ 0 \ \overline{\wedge} & 1 \\ 1 & 1 \\ & 0 \end{bmatrix} = \begin{bmatrix} \overline{X_1X_3} & X_1X_3 \\ 1 & 00 \\ 1 & 01 \\ 1 & 10 \\ 0 & 11 \end{bmatrix} \overline{\wedge}_{\times}$$

$$\overline{\wedge}_{\times} \begin{bmatrix} \overline{X_2\overline{X_3X}_4} & X_2X_3X_4 \\ 1 & 000 \\ 1 & 001 \\ 1 & 010 \\ 1 & 011 \\ 0 & 100 \\ 0 & 101 \\ 0 & 110 \\ 1 & 111 \end{bmatrix} = (0001\ 0010\ 0101\ 1111);$$

$$C_6 = \begin{bmatrix} & \overline{X_3X}_4 \\ X_2 & 1 \\ 0 \ \overline{\wedge}_{\times} & 1 \\ 1 & 1 \\ & 0 \end{bmatrix} \overline{\wedge}_{\times} \begin{bmatrix} \overline{X_3X}_4 & \\ 1 & X_5 \\ 1 & \overline{\wedge}_{\times} \ 0 \\ 1 & 1 \\ 0 & \end{bmatrix} =$$

$$= \begin{bmatrix} \overline{X_2\overline{X_3X}_4} & X_2X_3X_4 \\ 1 & 000 \\ 1 & 001 \\ 1 & 010 \\ 1 & 011 \\ 0 & 100 \\ 0 & 101 \\ 0 & 110 \\ 1 & 111 \end{bmatrix} \overline{\wedge}_{\times} \begin{bmatrix} \overline{\overline{X_3X_4}X_5} & X_3X_4X_5 \\ 1 & 000 \\ 0 & 001 \\ 1 & 010 \\ 0 & 011 \\ 1 & 100 \\ 0 & 101 \\ 1 & 110 \\ 1 & 111 \end{bmatrix} =$$

$$= (0101\ 0100\ 0111\ 1110).$$

3. Minimization of the resulting functionality cube in order to obtain a more compact solution by superposition of the tables or exclude irrelevant variables.

In the above example, the synthesis of the functionality cube is performed in accordance with the order of increasing addresses of the input variables, provided the consistency of the values of the same variables in each row of received truth table. For obtaining compact truth table of functionality provided by the circuit structure, it is necessary to perform superposition of two tables, corresponding to the outputs:

$$Y_1 = C_5 = (0001\ 0010\ 0101\ 1111);$$
$$Y_2 = C_6 = (0101\ 0100\ 0111\ 1110)$$

We can obtain a solution in the form of a truth table, which in this case is not completely ordered at the addresses of variables for the second function, but it is a compact table of 16 rows, presented below:

X_1	X_2	X_3	X_4	X_5	Y_1	Y_2
0	0	0	0	0	0	0
0	0	0	1	0	0	0
0	0	1	0	0	0	0
0	1	0	0	0	1	0
1	0	0	0	1	0	1
0	0	1	1	1	0	0
0	1	0	1	0	1	1
1	0	0	1	1	0	1
0	1	1	1	0	0	1
1	0	1	1	0	1	0
0	1	1	0	0	0	1
1	0	1	0	1	1	1
1	1	0	0	1	1	1
1	1	0	1	1	1	1
1	1	1	0	1	1	1
1	1	1	1	1	1	0

What does the practice of obtaining the coverage in the form of a cube, which is a vector of output states of the functionality? 1. First of all it is easy to specify of arbitrarily complex circuit of n variables by a binary vector of the dimension 2^n. 2. Processability of implementation of this form of functionality in the chips of programmable logic, where logic functions are represented by truth tables, realized as address memory. 3. Processability of functionality synthesis through the use of LUT (Look Up Table) by dividing the vector-cube on segments multiple the power of two. It should be noted that the logic function of any structure is quite difficult to realize in regular memory elements without introducing redundancy. Therefore, the proposed approach for obtaining a single cube of the logic circuit and subsequent decomposition of the vector on segments, corresponding to the existing in PLD tabular LUTs (4 inputs) is an effective technological solution. 4. Compactness of writing functionality in the form of a vector for solving the problems of analysis, fault simulation and fault free behavior, test synthesis and fault detection. 5. Processability and effectiveness of addressed implementation of Boolean structures of n variables in the software models, where the address space does not need to divide on segments in compliance with the limitations of the hardware primitives. 6. Transformation of the cubic structure in the functionality cube and subsequent decomposition of it on regular segments corresponding to the tabular LUT-primitives greatly simplifies solving the tasks of prototype testing, implemented in the chip PLD. 7. The vector (cube) of outputs $C = (C_1, C_2, ..., C_i, ..., C_p), p = 2^n$, is implicit and compact (not analytical or tabular) representation of

Boolean function, focused on addressed executing of logic operations:

$$Y = f(X_1, X_2, ..., X_i, ..., X_n) = A(X_1, X_2, ..., X_i, ..., X_n) = \\ = C(X_1, X_2, ..., X_i, ..., X_n) = C(A_1, A_2, ..., A_i, ..., A_n).$$

For instance, the function of three variables $Y = X_1X_2 \vee X_3 = A(X_1, X_2, X_3)$ after synthesis of the output vector $C = (C_1, C_2, ..., C_i, ..., C_p), p = 2^3$ will be represented in the form of cube C = (01010111), where each bit defines the state of the functionality depending on the address (the address of the cell) formed by the input word of three bits, corresponding to the variable $Y = A(X_1, X_2, X_3)$. For example, $Y = C(A = 011) = 1$. If the length of the cube C is known and is equal to p, then the number of variables forming the address for the extraction of required bit from C-vector is given by: $n = \log_2 p$.

4. Implementation of the functionality in the technological PLD structure

Solving the problem reduces to the partition of the functionality cube on segments, which can be associated with library primitives of the chip. There are some four-input primitives (memory cells for storing the truth table LUT) on chip PLD (FPGA). Primitives can be packed in logical sectors (Slice), consisting of two primitives, combined by multiplexer that allows increasing the digit capacity of input variables up to five. Subsequent union of two sectors through the multiplexer increases the digit capacity of input variables up to six. At that chip technology provides that every two pairs of sectors are multiplexed in blocks CLB (Configurable Logic Block), which makes it possible to increase the number of inputs of the synthesized logic function within CLB up to seven variables. The structure of the above modules (LUT, Slice, CLB) is shown in Fig. 2.

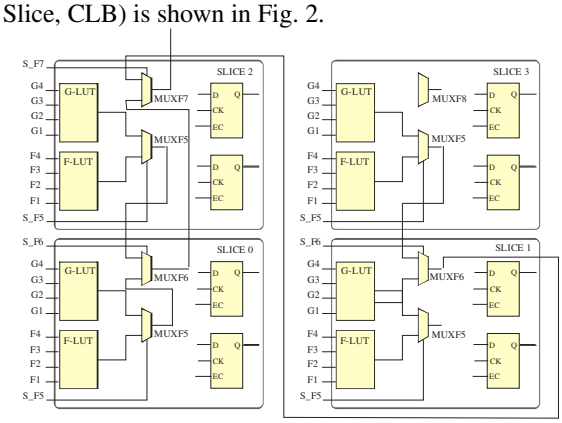

Fig. 2. The structure of PLD models

Here, the function value of seven variables is formed at the output of the multiplexer MUXF7. The subsequent increase of the digit capacity of input variables is associated with forced structural organization of CLB components in the hierarchical structures of higher level: one CLB is capable of forming any function of seven variables, 2 CLBs – 8 variables, 4 CLBs – nine variables, eight CLBs – 10 variables. In general, the functional dependence of the quantity of variables n upon the number of CLB blocks N has the following form:

$$n = \log_2 N + 7.$$

To estimate the hardware complexity of synthesis or implementation of computational structure it is necessary to have an inverse relationship of quantity of logical blocks CLB upon the complexity of logic function, reduced to the number of variables n or power of functionality cube $N = 2^n$. For example, for the function of n = 10 variables, it is necessary to have 8 CLBs on-chip. In general, the number of logical CLB blocks, depending on the quantity of variables of the synthesized function, is equal to:

$$N = 2^{n-7}, \quad n = 7,8,9, \ldots .$$

As the number of LUT primitives, then the estimation of the hardware complexity will be represented by the following expression:

$$N^* = 8L \times 2^{n-7} + 4M \times 2^{n-7} = (8L + 4M) \times 2^{n-7}.$$

Here 8L, 4M are hardware complexity of LUT implementation and multiplexor, included in CLB. At that the important characteristics of functionality implementation – the structural depth as the number of hierarchy levels, expressed in the number of LUTs and/or multiplexers on the longest logical path, and its delay D – also depend on the number of variables (delay of the multiplexer):

$$S = (n - 3);$$
$$D = (n - 3) \times D_M.$$

An instance: a cube of digital structure contains $p = 4096$, $(n = 12)$ bits (variables).

After synthesis the characteristics of such functionality will be the followings:

$$N = 2^{12-7} = 32;$$
$$N^* = (8L + 4M) \times 2^{n-7} = (8L + 4M) \times 2^5 = 32 \times (8L + 4M);$$
$$S = (12 - 3) = 9;$$
$$D = 9 \times D_M.$$

If the temporal characteristics of primitives for a particular chip of programmable logic, as well as hardware cost estimates of implementation of the structural components (LUT, Slice, CLB) are known, the parameters of functionality, implemented in the chip PLD (FPGA), can be calculated more accurately.

4. Conclusion

The scientific novelty lies in the proposed superposition method for synthesizing the cube of functionality, which is characterized by the use of the structure of primitive elements. For specific class of devices it can considerably reduce the time of obtaining the model of digital device, focused to the implementation in PLD chips.

The practical significance consists of significant increase in the speed of synthesis and minimization of cubic coverage of logic models through parallel execution of vector logic operations and taking into account the structural features of programmable logic devices. The method for constructing the functionality cube makes it possible to simplify solving the synthesis problem for the digital structure on chip through the use of components (LUT, Slice, CLB), which is of particular interest for its industrial use.

10. References

[1] Beth T. Quantum computing: an introduction // Proceedings of the IEEE International Symposium on Circuits and Systems, ISCAS.– 2000.– Geneva.– Vol. 1.– P. 735 – 736.

[2] Jonker P., Jie Han. On quantum and classical computing with arrays of superconducting persistent current qubits // Proceedings of Fifth IEEE International Workshop on Computer Architectures for Machine Perception.–2000.– P. 69 – 78.

[3] Keyes R.W. Challenges for quantum computing with solid-state devices // Computer.– Jan. 2005.– Vol. 38, Issue 1.– P. 65 – 69.

[4] Glassner A. Quantum computing. 3. // IEEE Computer Graphics and Applications.– Nov/Dec 2001.– Vol. 21, Issue 6.– P. 72 – 82.

[5] M.F. Bondaryenko, O.A. Guz, V.I. Hahanov, Yu.P. Shabanov-Kushnaryenko. Infrastructure for brain-like computing. Kharkov: Novoye Slovo. 2010. 160 p.

[6] Mark Gregory Whitney. Practical Fault Tolerance for Quantum Circuits. PhD Dissertation in Computer Science. Berkeley: University of California. 2009. 206p.

[7] Hahanov V. I., Litvinova E. I., Chumachenko S. V., Guz O.A. Logic associative computer. Electronic modeling. 2011. No 1. P. 73-90.

[8] Hahanov V., Wajeb Gharibi, Litvinova E., Chumachenko S. Information analysis infrastructure for diagnosis. Information. An international interdisciplinary journal. 2011. Japan. Vol.14. No 7. P. 2419-2433.

[9] V.I. Hahanov. “Digital System-on-Chip Design and Test. Kharkov: Novoye Slovo, 2009. 484 p.

Camera-ready was prepared in Kharkov National University of Radio Electronics
Lenin Ave, 14, KNURE, Kharkov, 61166, Ukraine

Approved for publication: 27.08.2012. Format 60×841/8.
Relative printer's sheets: 49. Circulation: 200 copies.
Published by SPD FL Stepanov V.V.
Ukraine, 61168, Kharkov, Ak. Pavlova st., 311

Матеріали симпозіуму «Схід-Захід Проектування та Діагностування – 2012»
Макет підготовлено у Харківському національному університеті радіоелектроніки
Редактори: Володимир Хаханов, Світлана Чумаченко, Євгенія Литвинова
Пр. Леніна, 14, ХНУРЕ, Харків, 61166, Україна

Підписано до публікації: 27.08.2012. Формат 60×84$^{1}/_{8}$.
Умов. друк. Арк. 49. Тираж: 200 прим.
Видано: СПД ФЛ Степанов В.В.
Вул. Ак. Павлова, 311, Харків, 61168, Україна