Checkable FPGA Design: Energy Consumption, Throughput and Trustworthiness

Alex Drozd[1], Svetlana Antoshchuk[1], Julia Drozd[1], Konstantin Zashcholkin[1], Miroslav Drozd[1], Nikolay Kuznietsov[1], Mohammed Al-Dhabi[1], Valery Nikul[1]

[1] Odessa National Polytechnic University. 1, Shevchenko avenue, Odessa Ukraine
drozd@ukr.net, asgonpu@gmail.com, dea_lucis@ukr.net, const-z@te.net.ua, miroslav_dr@mail.ru, koliaodessa@mail.ru, leraniku@gmail.com, aldhabi@mail.ru

1 Introduction

1.1. Motivation

FPGA (Field Programmable Gate Array) design is an important area in Industrial Application of Green IT-Engineering, which is positioned in two main directions: energy efficiency and safety [1, 2]. Energy efficiency characterizes a degree energy quantum use [3, 4], and safety determines the direction in which energy would be expended: for the creation or the corruption. [5, 6] These directions are tightly connected in the areas of critical application. The objects of the increased risk which provide production, transmission and consumption of the great amount of electric power, have already become a natural human environment. These objects are becoming more complex and more powerful. Their number is growing. The equilibrating task of the increasing risks is defined for Green IT, which is implemented in instrumentation and control safety-related systems aimed at providing of the functional safety of both the system and the object for accident prevention and of their consequences reducing [7]. The cascade power networks disconnections and accidents at the power stations lead to the considerable electric power losses as a result of stoppings in its delivery and production. The important feature of the modern stage of Green IT development is creation of safety-related systems on FPGA [8]. This fact represents the high evaluation of FPGA opportunities design.

FPGA design supports all types of circuit parallelism: matrix, pipeline and preparation of results which are selected with the use of the resource-based approach at the levels of resource development: replications, diversifications and autonomy, accordingly [9]. However, the matrix parallelism, reflecting the bottom level of resource development is the dominant one [10, 11]. It is represented not only in the matrix organization of chips, but, first of all, in orientation of design to processing of numbers in parallel codes.

The matrix parallelism demonstrates a lot of drawbacks inherent in the bottom level of resources development. Array structures occupy big spaces, but do not provide the appropriate temporal parallelism. The iterative array multiplier of n-bit binary codes executes the operation for one clock period and for this purpose contains n^2 operational elements, $2n - 2$ of which are connected consistently. Therefore, each operational element is used in a clock period for a short period of time with the coefficient of $K_M = (2n - 2)^{-1}$ [12]. For $n = 32$ and $n = 64$, operational elements are used only for 1.6% and for 0.8% of time, accordingly. The iterative array divider contains $(n + 1)^2$ sequentially connected operational elements. For each of them it defines utilization coefficient $K_D = (n + 1)^{-2,}$ i.e. for $n = 32$ and $n = 64$ operational elements are used for 0.1% and 0.02%, accordingly. The rest of the time in a clock cycle it is used for the distribution of waves of parasitic transitions, the amount of which repeatedly exceeds the number of the functional ones. This glitch problem leads to the significant increase in a dynamic component of energy consumption [13, 14]. Input, processing and output of data in parallel codes causes a large number of connections in array circuits, which makes their topology more complicated, reduces technological effectiveness in production and reliability in maintenance.

Therefore, we suggest to develop Green IT on the basis of array structures reduction, suggesting two ways: The Soft Way and Cardinal one.

The Soft Way consists in execution of the truncated arithmetical operations which almost twice simplify array circuits and reduce time of computation with maintaining single precision when the result inherits the size of an operand [15]. Therefore, the truncated operations executed with mantissas are the main method of processing the approximate data in floating-point formats [16]. The resource approach represents a tendency of the amplifying domination of the approximate calculations over the exact data processing [9, 17]. In the safety-related systems, the initial data arriving from sensors is a result of measurements, i.e. also approximate data.

The Cardinal Way presupposes the transition to the following level of the resources development: from replication of array structures to diversification on the basis of pipeline enhancement. Modern computer systems and their digital components, as a rule, are built like a pipeline which reflects the diversification level. However, pipeline sections are single-cycle nodes with array structures and their problems. We suggest to reduce array structures of the pipeline sections. The maximum reduction of array structure to one operational element will transform the system into the bitwise pipeline for data processing in consecutive codes. To what extent such progressive decision can be competitive in the world of array structures?

Decades of matrix parallelism domination promoted the considerable achievements in its enhancement that substantially strengthened the position of matrix decisions. FPGA design uses these achievements effectively. Preparation of results is used for an increase in array structures efficiency: circuits of the accelerated distribution of add carries of parallel codes are prepared; iterative array multipliers are prepared with high indexes in throughput and energy saving; libraries suggest the wide range of nodes with array structure [18].

Can a cardinal solution compete with array structures in these conditions?

The affirmative answer is based on one indisputable advantage of bitwise pipelines in comparison with array structures. This advantage consists in the high checkability which is very important for the safety-related systems.

These systems are designed for operation in two modes: normal mode and emergency one. The functional safety is ensured by the use of fault-tolerant decisions which together with dual-mode operation of systems lead to the considerable structural redundancy of digital circuits and, as a result, to a low checkability of their array structures. It creates a problem of the hidden faults which in a normal mode are not detected as there are no input data necessary for this purpose. The continuous normal mode adds to the accumulation of the hidden faults. In the most responsible emergency mode such faults are represented, they reduce fault tolerance of components and the functional safety of systems. Points of a circuit in which this problem is represented belong to potentially hazardous ones [19, 20].

The Problem of the Hidden Faults is known by unsuccessful attempts to solve it with the use of the imitation modes. The accident simulation executed in a case of a switch-off emergency protection repeatedly led to the alert conditions. The Chernobyl catastrophe took place in the case of the turned-off emergency protection.

Register structures of bitwise pipelines are the elements of testable design (scan registers). The serial codes propagated in circuit by scan registers that do not leave place for hidden faults. Therefore, bitwise pipelines deserve an attention directed to the receiving of their estimates in comparison with array structures and development of models, methods and means for their enhancement.

FPGA design represents additional opportunities of an increase in the functional safety of safety-related systems on the basis of reconfigurable components creation, thanks to high version redundancy of circuits in LUT-oriented architecture (LUT – Look-Up Table) [21, 22]. The version redundancy allows to select the most effective configuration of a component at a stage of its design, and also in the process of FPGA project program code forming [23].

1.2. Related works

The known works related to the problems of green FPGA design can be subdivided into three groups which show:

- The possibilities of FPGA design that allow to carry out comparative assessment of matrix circuits and bitwise pipelines;
- The development of on-line testing methods for circuits with the reduced array structures;
- The use of version redundancy of circuits for increase in safety of the digital components implemented in FPGA projects.

The possibilities of FPGA design are considered on the example of CAD Altera [24]. For the comparative analysis of matrix and pipeline circuits in energy consumption, utilities of Power Play Early Power Estimators Spreadsheets and Power Play Power Analyzer described in [25, 26] are used. An assessment of temporal parameters is executed by temporal simulation of circuits with the use of Timing Analyzer utility [27].

The review of the on-line testing methods which increase their trustworthiness in the approximate results checking allows to select the residue checking methods and methods checking results by inequalities [28]. The important feature of approximate data consists in their non-uniform structure which contains most and least significant bits [29]. Faults in circuits cause the errors in these bits, they are accordingly essential and inessential for the result trustworthiness.

The residue checking methods developed for the truncated operations executed in iterative array multipliers and dividers of mantissas are effective in case of high probability of an essential error [30]. The operations of a denormalization of operands and normalization of results considerably reduce this probability for results of all prior and all following operations with mantissas, accordingly. In these conditions residue checking methods reduce the trustworthiness as a result of ignoring of approximate data structure as they do not distinguish between an error in most and least significant bits [31].

The methods of checking results by inequalities show high trustworthiness, distinguishing between essential and inessential errors and they do not depend on a method of obtaining result. It allows to use them directly for checking of the truncated operations. However, these methods are not developed for the main formats regulated by the standards IEEE Standard 754-1985 and IEEE Standard 754™-2008 for floating-point arithmetic [32, 33].

The version redundancy inherent to FPGA is represented in an opportunity to have a set of versions of a program code for the same hardware implementation of the circuit decision in the LUT-oriented architecture. The versions are formed in the case of the data transfer from one LUT to another in a direct or inverse value. Use of such version redundancy became a basis for an increase in the functional safety of FPGA projects by an increase in a checkability of the circuit in a normal mode and trustworthiness of results in the emergency one [23].

The version redundancy of the program code allows to execute monitoring of FPGA project integrity, too. The integrity of the information object (IO) is a part of security which can be considered as a necessary condition in the functional safety support.

The simplest scheme for the monitoring integrity determines a digest, which presupposes a standard hash sum with hash function usage, for IO [34]. The IO integrity is considered to be confirmed if a calculated hash sum coincides with a standard one. The disadvantages of this scheme are the following: 1) A standard hash sum is exposed; 2) The hash sum is physically separated from IO; 3) The fact of IO integrity is being controlled cannot be actually concealed from the outside surveillance. All these factors make it possible for the integrity violation to be concealed by substituting the hash sum.

The drawbacks 1 and 2 can be changed by encrypting of the hash sum and by attaching this procedure result to the initial IO [35]. To monitor the integrity, the controlling information H is separated from verified IO A. As a result, the information object acquires the state A_0, which it had before attaching the controlling part. The hash sum for object A_0 is compared to the hash sum H earlier subjected to the decryption. The decision of the object integrity is made according to the results of this comparison. The disadvantages of this decision are the following:

4) Complexity or even impossibility of the direct insertion of the encrypted hash sum into the active IO body; 5) Disclosure of the fact of the controlling information existence which gives the possibility of breaking the encrypted data (brute force attack, complex attack, social engineering).

The approach allows to attach the controlling information to the information object not with the help of mere concatenation, but is known in the form of digital watermark (DWM). This method is able to conceal the fact of controlling information existence in the object. Methods and approaches of this type [36, 37] are thoroughly studied for passive IOs. The experience of the passive IO integrity provision can be used in developing of active IO integrity control methods and exactly the program code of FPGA.

1.3. Goal and structure

The purpose of the researches is to develop Green Technologies on the basis of array structures reduction and the use of version redundancy of FPGA projects for an increase in the safety of the digital components in the areas of critical application.

Section 2 represents the comparative analysis of matrix circuits and bitwise pipelines in complexity, throughput and energy consumption for FPGA projects. The received estimates allow to define ways of increasing the bitwise pipelines efficiency. The method of throughput increase and lowering in energy consumption for the bitwise pipeline multiplier by processing two bits of operands in a clock cycle is offered.

Section 3 suggest on-line testing methods for digital components safety increase in normal and emergency modes. These methods are oriented on checking of the truncated arithmetical operations, implemented in the reduced array structures with the raised checkability which is important for normal mode. Methods execute checking of mantissas by inequalities which increase trustworthiness of on-line testing in emergency mode by distinguishing of essential and inessential errors. Methods are based on features of standards IEEE Standard 754-1985 and IEEE Standard 754™-2008.

Section 4 represents version redundancy of the program code in the FPGA project for a solution of problem of the hidden faults.

Section 5 describes the method of integrity monitoring of the FPGA project on the basis of version redundancy of its program code.

2 Bitwise pipelines vs. Array structures

2.1. Comparative analysis of matrix circuits and bitwise pipelines

The high checkability of bitwise pipelines which cardinally solves a problem of the hidden faults, raises questions about other factors of efficiency in comparison with traditionally used array structures. To the following factors belong the complexity, throughput and energy consumption. The question of the relative efficiency of bitwise pipelines is raised in connection with unequal conditions of array and bitwise structures implementation in FPGA projects. The modern CAD (Computer-Aided Design) first of all is oriented on the array structures support.

Comparison between an array structure and a bitwise pipeline is executed experimentally on the example of multiplication operation with the use of a CAD Altera Quartus II v13 64 bit and FPGA Altera of the Cyclone II EP2C35F672C6 family [38]. Throughput is evaluated in the temporal parameters received by means of the utility TimeQuest Timing Analyzer [27], and energy consumption is calculated depending on the power determined by the utility of Power Play [25, 26]. Throughput and energy consumption are compared in a case of identical circuits complexity: one iterative array multiplier is compared to several pipelines of the same summary complexity. The complexity is estimated by the number of the LE (Logic Elements).

Table 1 represents the results of experiments for the size of operands $n = 8$ and $n = 16$.

Table 1 Results of comparison between iterative array and bitwise pipeline multipliers

Index	n	Array structure	Bitwise pipeline	Effectiveness
Complexity, amount of LE	8	179	159 (5 pipelines)	–
	16	739	626 (10 pipelines)	–
Delay of clock cycle, ns	8	8.069	2.38	–
	16	17.238	2.38	–
Throughput, amount of operations / s	8	$124 \cdot 10^6$	$148 \cdot 10^6$	1.19
	16	$58 \cdot 10^6$	$155 \cdot 10^6$	2.67
Power, mW	8	6.37	26.11	–
	16	11.24	77.51	–
Energy consumption, mW·ns	8	51.4	198.9	0.26
	16	193.8	590.3	0.33

Results of experiments show advantage of bitwise pipelines in throughput of 1.19 and 2.67 times and loss in energy consumption in 4 and 3 times (the relative efficiency – 0.26 and 0.33) for the size of operands $n = 8$ and $n = 16$, accordingly. The relative efficiency of bitwise pipelines is increased with the growth of the operands size.

Specific energy consumption, i.e. in the terms of one 8-bit multiplication, for iterative array multipliers is defined as $K_M = P\ T / N$, where P – power consumption, T_M – runtime of one operation in the iterative array multiplier, $N = (n / 8)^2$ – the equivalent number of 8-bit operations [39].

Specific energy consumption for bitwise pipeline multipliers is defined as

$$K_P = P_P\ T_P / (N\ C),$$

where P_P – the power consumed by C pipelines which in the amount have complexity of the iterative array multiplier; $P_P = P\ T / \tau$, τ – pipeline clock cycle;
$C = C_M / C_P$, C_M and C_P – complexity of array and bitwise pipeline multipliers;
$T_P = Z\ \tau$ – runtime of one operation by the pipeline;
Z – amount of clock cycles in pipeline operation, $Z = 2n\ \tau$.

Then $K_P = (P\ T / \tau)\ (Z\ \tau\ ) / (N\ C) = P\ T / N\ Z / C) = K_M\ \ Z / C$ and this equality allows to evaluate relative efficiency $K_{P\,/\,M} = K_M / K_P$ of the bitwise pipeline in energy consumption as $K_{P\,/\,M} = C / Z$. Thus, the relative energy efficiency $K_{P\,/\,M}$ of the pipeline multiplier depends only on relative complexity of array structure and the amount of clock cycles in a pipeline operation.

2.2. A method of increase in energy efficiency of the bitwise pipeline multiplier

Based on the received assessment, we suggest a method of energy efficiency increase of the bitwise pipeline multiplier by reduction of Z, processing in each clock cycle of two operands bits. Multiplier calculates 4 groups of conjunctions: 1.1 – 1.4, 2.1 – 2.4, 3.1 – 3.4 and 4.1 – 4.4 shown in Table 2 for $n = 8$.

Table 2 Computation of conjunctions on clock cycles and weights of product bits

Clock cycle	1		2		3		4		5		6		7		8	
Weights of bits	2^0	2^1	2^2	2^3	2^4	2^5	2^6	2^7	2^8	2^9	2^{10}	2^{11}	2^{12}	2^{13}	2^{14}	2^{15}
1.1	11		13		15		17									
1.2			31		33		35		37							
1.3					51		53		55		57					
1.4							71		73		75		77			
2.1		21		23		25		27								
2.2				41		43		45		47						
2.3						61		63		65		67				
2.4								81		83		85		87		
3.1			22		24		26		28							
3.2					42		44		46		48					
3.3							62		64		66		68			
3.4									82		84		86		88	
4.1		12		14		16		18								
4.2				32		34		36		38						
4.3						52		54		56		58				
4.4								72		74		76		78		

Conjunctions are designated by the codes that consist of the operands numbers. Conjunctions 3.1 – 3.4 of the third group after computation are delayed on one clock cycle as it is represented in the Table 2.

We compare bitwise pipeline multipliers 1 and 2, processing 1 and 2 bits in clock cycle, accordingly. Table 3 represents the results of the experiments for the size of operands $n = 8$ and $n = 16$.

Table 3 Results of comparing the pipeline multipliers 1 and 2

Index	n	Bitwise pipeline 1	Bitwise pipeline 2	Effectiveness
Complexity, amount of LE	8	159 (5 pipelines)	182 (4 pipelines)	–
	16	626 (10 pipelines)	625 (6 pipelines)	–
Delay of clock cycle, ns	8	2.38	2.38	–
	16	2.38	2.38	–
Throughput,	8	$148 \cdot 10^6$	$184 \cdot 10^6$	1.24

amount of operations / s	16	$155 \cdot 10^6$	$158 \cdot 10^6$	1.02
Power,	8	26.11	26.06	–
mW	16	77.51	56.80	–
Energy consumption,	8	198.9	124.0	1.60
mW·ns	16	590.3	360.5	1.64

Results of experiments show advantage of bitwise pipeline processing 2 bits in clock cycle by suggested method in throughput in 1.24 and 1.02 times and in energy consumption in 1.60 and 1.64 times for the size of operands $n = 8$ and $n = 16$, accordingly.

3 On-line testing methods for digital components safety increase

3.1. Requirements to on-line testing of safety-related systems

On-line testing of digital components of safety-related systems inherits their features which determine requirements to checking of result trustworthiness. Diversification of an operating mode of safety-related systems on normal and emergency modes leads to diversification of these requirements determining a directivity of on-line testing in these modes. In a normal mode the safety-related system is waiting for an emergency mode and is directed on maintenance of a high level of readiness to anticipate accident and to reduce its consequences. Therefore, on-line testing performs functions of the testing that are not absolutely peculiar to it and aimed at fault detection. However, testing is executed with the use of tests. On-line testing is using uncontrollable input sequences of operating words and completely depends in the trustworthiness on these words in their ability to become tests.

For on-line testing the checkability of circuits becomes the functional, i.e. represents dependence on input data, and for safety-related systems – dual-mode [19. 20].

At the same time, the high checkability of circuits leads to more frequent manifestation of the errors caused by the transient faults in the emergency mode. These errors, as a rule, are inessential for trustworthiness of the approximate results peculiar to safety-related systems, however they reduce trustworthiness of on-line testing, distracting its resources in emergency mode on their parrying. The three-channel majority system will correctly continue to function in a case of a permanent fault of one channel, but transient faults against the background of this permanent fault will lead to incorrect results.

The problem of different requirements to on-line testing in the normal and emergency mode needs to be solved in a complex, raising a checkability of circuits and weakening influence of inessential errors. Such decision shall be based on the development of on-line testing methods which at the same time meet two conditions: they allow to service circuits with the reduced array structures and distinguish essential and inessential errors in approximate results.

Methods checking mantissas by inequalities completely meet these requirements. They are oriented on checking in trustworthiness of n-bit results of homogeneous arithmetical operations with n-bit operands and do not depend on a method of obtaining result which can be calculated by the circuits with the reduced array structure. At the same time, these methods evaluate the size of an error and detect essential errors more likely in comparison with inessential errors.

3.2. Checking mantissas by inequalities in truncated operations

The method of mantissas checking by inequalities determines the lower and upper boundaries of a result calculated on operands and compares the result with these boundaries. The result is considered to be non-authentic if it exceeds an upper bound or if it appears below the lower bound [28].

The boundaries are created of the restrictions which are superimposed on mantissas of the normalized floating-point numbers in the formats regulated by IEEE Standard 754-1985 and IEEE Standard 754™-2008 [32, 33].

For mantissas multiplication operation $A \cdot B = V$, the operands are restricted to inequalities $1 \leq A < 2$ and $1 \leq B < 2$, which are integrated in the following conditions: $(A - 2)(B - 2) > 0$ and $(A - 1)(B - 1) \geq 0$, defining lower bounds of the product: $V_{\mathrm{LM.1}} = 2((A + B) - 2)$, $V_{\mathrm{LM.2}} = A + B - 1$ and $V_{\mathrm{LM.1\text{-}2}} = \max(V_{\mathrm{LM.1}}, V_{\mathrm{LM.2}})$, where $V_{\mathrm{LM.1}} < V$, $V_{\mathrm{LM.2}} \leq V$ and $V_{\mathrm{LM.1\text{-}2}} \leq V$. Similarly, conditions $(A - 1)(B - 2) \leq 0$ and $(A - 2)(B - 1) \leq 0$ define upper bounds of the product: $V_{\mathrm{HM.1}} = 2A + B - 2$, $V_{\mathrm{HM.2}} = A + 2B - 2$ and $V_{\mathrm{HM.1\text{-}2}} = \min(V_{\mathrm{HM.1}}, V_{\mathrm{HM.2}})$, where $V_{\mathrm{HM.1}} \leq V$, $V_{\mathrm{HM.2}} \leq V$ and $V_{\mathrm{HM.1\text{-}2}} \leq V$. Lower bounds $V_{\mathrm{LM.1}}$ and $V_{\mathrm{LM.2}}$ serve as a basic for combined lower bound $V_{\mathrm{LM.1\text{-}2}}$. The upper bounds of $V_{\mathrm{HM.1}}$ and $V_{\mathrm{HM.2}}$ are basic for combined upper bound $V_{\mathrm{HM.1\text{-}2}}$ [40].

The described 3 upper and 3 lower bounds allow to make 9 methods for checking mantissas by inequalities. The trustworthiness of methods is defined by an ability of boundaries to correctly distinguish essential and inessential errors.

The methods are evaluated in the trustworthiness by experiments with the use of program model of the iterative array multiplier. The program model is developed on a demo version of Embarcadero Delphi 10.1 [41] for an assessment in trustworthiness of mantissas checking methods by inequalities. Simulation is executed for two $\mathrm{M_B}$ and $\mathrm{M_C}$ methods which use pairs of basic boundaries $V_{\mathrm{LM.1}}$, $V_{\mathrm{HM.1}}$ and pair of combined boundaries $V_{\mathrm{LM.1\text{-}2}}$, $V_{\mathrm{HM.1\text{-}2}}$.

Experiments are made on random sequences of input data – mantissas of the normalized operands. Operation of truncated multiplication is executed under the influence of single faults like shorts between two points of the circuit within one operational element. Points of the circuit and an operational element are selected randomly.

Experiments separate errors into essential and inessential specifying quantity n_{MSB} of most significant bits in the product of mantissas.

Simulation allows to determine probability P_E of an essential error, detection probabilities P_{DE} and P_{DI} of essential and inessential errors, trustworthiness $D_{B.n}$ of the $\mathrm{M_B}$ method or trustworthiness $D_{C.n}$ of the $\mathrm{M_C}$ method, where n is the mantissa size. Calculated trustworthiness are compared to trustworthiness D_R of residue checking method which in an experiment showed the detection of all errors, i.e. its probability of error detection $P_D = 1$, and trustworthiness $D_R = P_E$. The result of comparing is determined for the M_B and M_C methods as $\delta_{B.n} = D_{B.n} / D_R$ and $\delta_{C.n} = D_{C.n} / D_R$.

Results of simulation of the M_B method for the size of mantissas $n = 8$ are shown in Table 4.

Table 4 Results of simulation of the M_B method for $n = 8$

n_{MSB}	3	4	5	6	7	8
P_E, %	6.7	12.9	20.0	31.4	44.5	58.5
P_{DE}, %	100	81.8	64.9	50.0	39.0	31.7
P_{DI}, %	12.7	10.7	7.3	4.5	3.2	1.4
$D_{B.8}$, %	87.3	88.1	86.4	80.4	70.5	59.0
$\delta_{B.8/R}$	12.9	6.8	4.3	2.6	1.6	1.0

The received results show an increase in probability P_E with growth of the number n_{MSB}. Lowering of n_{MSB} leads to growth of P_{DE}, P_{DI}, $D_{B.8}$ and $\delta_{B.8}$ indexes,

Results of simulation of the M_B method for the size of mantissas $n = 15$ are shown in Table 5.

Table 5 Results of simulation of the M_B method for $n = 15$

n_{MSB}	5	7	9	11	13	15
P_E, %	6.7	12.3	21.6	34.1	48.6	66.1
P_{DE}, %	66.7	42.1	24.2	17.3	13.3	9.7
P_{DI}, %	2.0	0.1	0	0	0	0
$D_{B.8}$, %	95.4	91.5	82.8	71.3	57.9	40.2
$\delta_{B.8/R}$	14.2	7.4	32.8	2.1	1.2	0.6

The high trustworthiness of the method is reached due to the low detection probability P_{DI} of an inessential error and increases with growth of detection probability P_{DE} of an essential error.

Results of simulation of the M_C method for the size $n = 8$ are shown in Table 6.

Table 6 Results of simulation of the M_C method for $n = 8$

n_{MSB}	3	4	5	6	7	8
P_E, %	6.7	12.9	20.0	31.4	44.5	58.5
P_{DE}, %	100	100	95.1	90.9	84.9	82.2
P_{DI}, %	74.5	72.9	71.5	70.4	68.9	67.1
$D_{B.8}$, %	30.5	36.5	42.8	48.9	55.5	61.5
$\delta_{B.8/R}$	4.6	2.8	2.1	1.6	1.2	1.0

The trustworthiness of the method grows together with the probability P_E of an essential error, thanks to the high detection probability P_{DE} of an essential error and the decreasing detection probability P_{DI} of an inessential error.

Results of simulation of the M_C method for the size of mantissas $n = 15$ are shown in Table 7.

Table 7 Results of simulation of the method M_C for $n = 15$

n_{MSB}	5	7	9	11	13	15
P_E, %	6.7	12.3	21.6	34.1	48.6	66.1
P_{DE}, %	100	88.9	80.4	76.4	73.5	71.9
P_{DI}, %	68.2	67.5	67.0	66.7	66.2	66.0
$D_{B.8}$, %	36.9	40.7	43.6	48.5	53.3	58.4
$\delta_{B.8/R}$	5.5	3.3	2.0	1.4	1.1	0.9

The trustworthiness of the method grows together with the probability P_E of an essential error in a case of insignificant lowering in detection probabilities P_{DE} and P_{DI} of an essential and inessential error.

The M_B and M_C methods are alternative, they increase trustworthiness with reduction and growth in probability P_E of an essential error, accordingly.

Comparison of trustworthiness of the M_B and M_C methods with the residue checking represents an advantage of checking mantissas by inequalities and this advantage grows with reduction in probability P_E of an essential error.

4 The hidden faults in FPGA projects: problem and decision

The problem of the hidden faults has a continuation in connection with the development of digital components of the safety-related systems on FPGA with LUT-oriented architecture [18, 19]. It is necessary to mention that this problem is peculiar for the safety-related systems as they belong to the dual-mode type of systems. Systems with one operating mode do not face such problem as the hidden faults remain hidden throughout all operation and are not dangerous. Therefore, use of FPGA in the safety-related systems requires carrying out the analysis on their tolerance in relation to the hidden faults.

LUT is the generator of Boolean functions. In case of four inputs A, B, C and D, LUT generates a Boolean function of four variables. Values of this function are stored in 16-bit memory of LUT. Functioning of LUT is characterized by sets M_N and M_E of memory bits (their addresses at inputs A, B, C and D) to which addressing in normal and emergency mode is executed, accordingly.

In case of $M_E \subseteq M_N$ all bits of memory addressed in emergency mode are used in the normal one. It interferes with the accumulation of the hidden faults in the normal mode.

The set $M_{PH} = M_N \setminus M_E$, $M_{PH} \neq \varnothing$ of the memory bits used in LUT only in emergency mode is a basis for a set of potentially hazardous points of the circuit. The hidden faults can be accumulated in these bits in a normal mode and reduce fault tolerance of digital components and the functional safety of safety-related systems in an emergency mode.

As an example, the digital component calculating the Boolean function $F(X) = 1$ for X mod 3 = 0 and $F(X) = 0$ for remaining values of X where $X = \{x_5, x_4, x_3, x_2, x_1\}$, $X = 0 \div 31$ is taken. Input data change in the ranges R_N and R_E respectively in normal and emergency mode. Design of this digital component on FPGA ALTERA Quartus II defines the circuit, represented in Fig. 1.

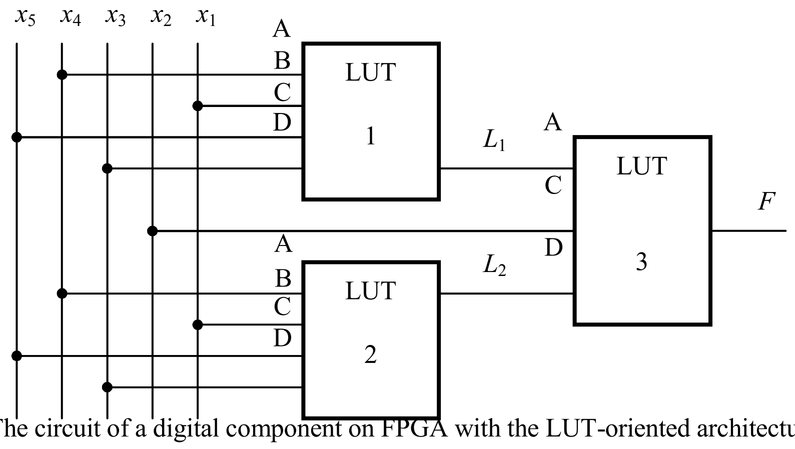


Fig. 1 The circuit of a digital component on FPGA with the LUT-oriented architecture

The circuit consists of three LUT: LUT 1, LUT 2 and LUT 3 which realize the functions $L_1(x_3, x_5, x_1, x_4)$, $L_2(x_3, x_5, x_1, x_4)$ and $F(L_2, x_2, L_1)$ described by codes $6BBD_{16}$, $97E9_{16}$ and $0AA0_{16}$, accordingly.

Values of the variables x_3, x_5, x_1, x_4 at the inputs of LUT 1 and LUT 2 and also corresponding to them values from the range X of the input data are shown in Table 8.

Table 8 Values at the inputs of LUT 1, LUT 2 and corresponding to them values of X

#	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
x_4	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
x_1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
x_5	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
x_3	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
X	0	8	1	9	16	24	17	25	4	12	5	13	20	28	21	29
	2	10	3	11	18	26	19	27	6	14	7	15	22	30	23	31

Each number # of the variables x_3, x_5, x_1, x_4 is created for two values of the range X in case of $x_2 = 0$ and $x_2 = 1$ accordingly.

We suggest three examples of partition of a set X of the input data on the ranges R_N and R_E of normal and emergency mode: a) $R_N = 0 \div 15$ and $R_E = 16 \div 31$; b) $R_N = 7 \div 26$ and $R_E = 0 \div 6, 27 \div 31$; c) $R_N = 0 \div 26$ and $R_E = 27 \div 31$.

The use of LUT memory (LUT 1 and LUT 2) in the normal and emergency modes is represented for the considered examples in Fig. 2, **a**, **b** and **c**, accordingly.

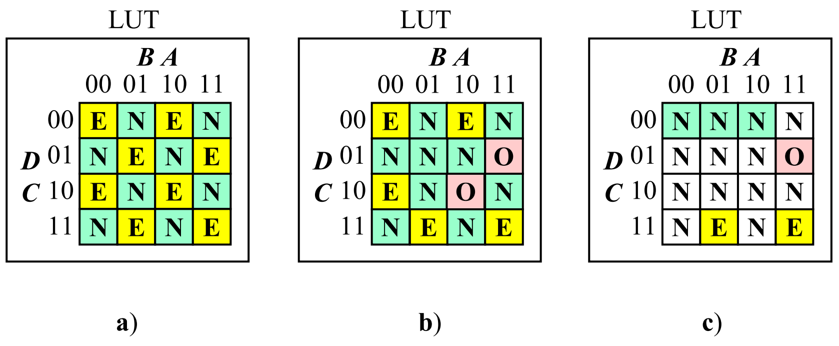


Fig. 2 The use of LUT memory in the normal and emergency modes

Memory of LUT contains the bits N, E and O, which are used only in normal, emergency or in the both modes, accordingly. The example of **a** shows only bits like E and N, i.e. $M_N \cap M_E = \varnothing$ and $M_{PH} = M_E$. The example of **b** contains also bits of O type. The hidden faults cannot be accumulated in bits of N and O types. The example of **c** represents an example of a wide range of values of the M_N set where only the first three bits are used. Then bits of O type can also accumulate the hidden faults as well as bits like E.

An example of **c** represents changes of input data in the insignificant range. In such cases the manual regulation of input data is used for the detection of the hidden faults. It allows to check bits of O type and to exclude them from a set of potentially hazardous points of the circuit. However manual regulation is powerless before a problem of the hidden faults for bits like E.

We suggest a method for the solving of this problem, using the version redundancy of the LUT-oriented architecture described in [23]. Two sequentially connected LUT allow to transfer between direct or inverse value without influence on a remaining part. The inverse value is created by an inverse of all bits in memory of the first LUT. Inverse on an input of the second LUT is compensated by the appropriate change of places of the bits in the memory. Such change concerns only the LUT program code and proves a number of its versions.

Inverting of A input or C input in examples **a** and **b** replaces all bits like E with bits of N type. Continued use of both versions (initial and changed) excludes a possibility of accumulation of the hidden faults in all LUT bits. Inverting of all LUT inputs interchanges the position of the first and last three bits of memory, solving a problem in the example of **c**. Manual regulation can be completely excluded by the continuous use of several program codes providing exchange of places for all bits like E with bits of N type.

5 The integrity monitoring of LUT-oriented IO

The given paper suggests a method to monitoring the integrity of FPGA program code, which is an active IO with the LUT-oriented architecture (hereinafter – LUT-object) [42]. The suggested method proves that the monitoring information is embedded in IO and is represented in the form of DWM. The integrity monitoring scheme within the framework of this method is represented in Fig. 3.

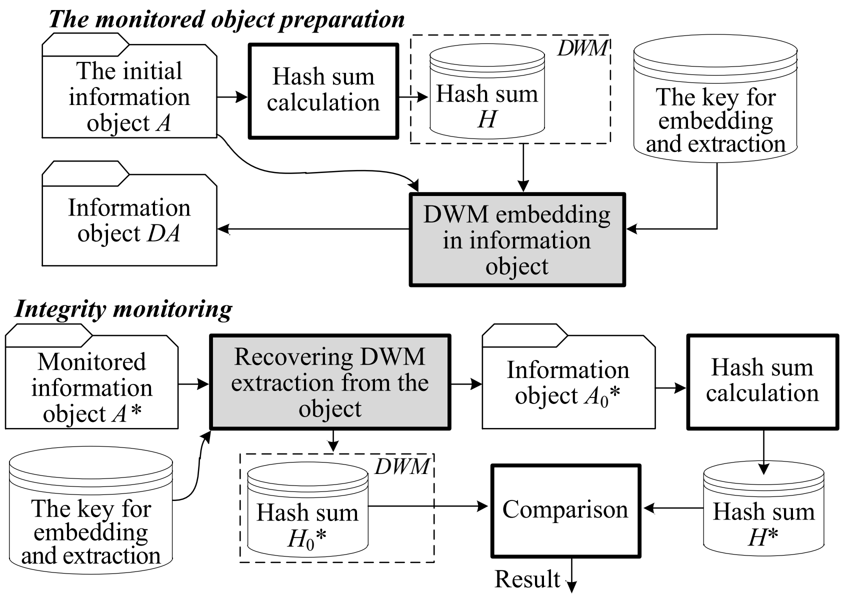

The monitored object preparation
The initial information object A
Hash sum calculation
DWM
Hash sum H
The key for embedding and extraction
Information object DA
DWM embedding in information object
Integrity monitoring
Monitored information object A*
Recovering DWM extraction from the object
Information object A_0*
Hash sum calculation
The key for embedding and extraction
DWM
Hash sum H_0*
Comparison
Hash sum H*
Result

Fig. 3 Integrity monitoring scheme used in the suggested method

The suggested method is based on the application of the following compositions: a) Fridrich-Goljan-Du's method (hereinafter the Fridrich's method) [43, 44] in the point that concern the provision of the object state recovery; b) method of the equivalent conversion of codes in LUT unit pairs [23].

The Fridrich's method is only used for the passive IOs and is based on the application of two functions: a) function of discrimination employed for classification of IO groups elementary parts by including them in one of three classes: regular R, singular S and unused U. The indication of the function value in each of the groups requires the complex calculation procedure; b) flipping-function used for the conversion of groups of IO elementary parts in accordance with some rule.

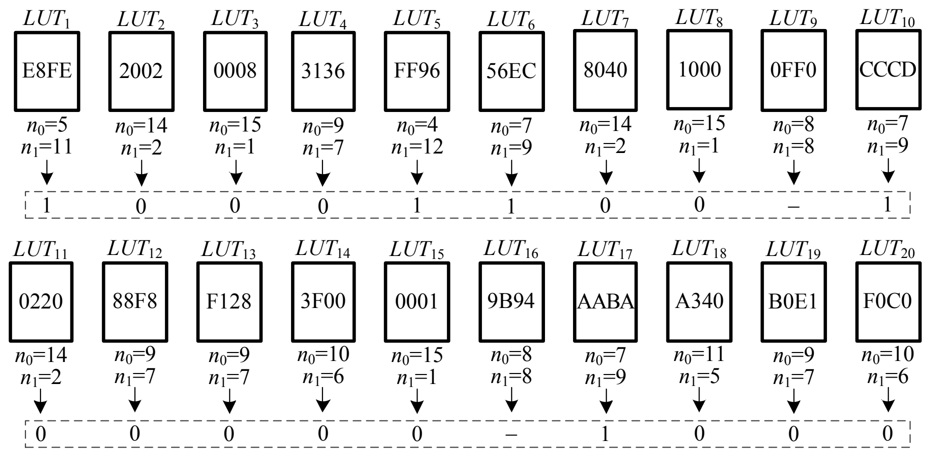
Unlike the Fridrich's method the approach suggested in the given paper proves that IO element classification is performed without discrimination function [45]. The classification is based on correlation of zero values and one values in the codes of LUT-units: if more zero values are present in the code then the unit is classified as a regular one (R-unit); if there are more one values in the code the unit is classified as singular (S-unit); if the amounts of zero and one values of the code coincide then the unit is classified as an unused one (U-unit).

Within the framework of our method the Flipping-function calculation, unlike the Fridrich's method, includes only the equivalent inversion of LUT unit values in accordance with the procedure suggested in [23]. As a result of the unit program, code inversion of the R-unit is converted into S-unit, S-unit is converted into R-unit, and

U-unit does not change its class. The sequence of actions in the point concerning the insertion of DWM with monitoring information into LUT-objects, which is suggested by the described method, may be represented in the following way.

Stage 1. The sequential examination of the LUT-object units in the order determined by the embedding path, which is an organized LUT unit subset obtained as a result of employing the preset rule of unit examination, is carried out. Wherein the classifying of units is performed. The result is represented as a binary *RS*-vector: value 0 is put for *R*-units, value 1 is put for *S*-units, and no values are put for *U*-units.

Fig. 4 shows an example of the embedding path containing 20 LUT_1–LUT_{20} units. The hexadecimal value of program code, the number of zero values n_0 and one values n_1 in this code, the result of classifying the unit and the *RS*-vector value is demonstrated for each of the units. Vector $RS = 100011001000001000$ consists of 18 bits since the information of the two *U*-units is not included in the *RS*-vector contents.



LUT_1 E8FE n_0=5 n_1=11 1
LUT_2 2002 n_0=14 n_1=2 0
LUT_3 0008 n_0=15 n_1=1 0
LUT_4 3136 n_0=9 n_1=7 0
LUT_5 FF96 n_0=4 n_1=12 1
LUT_6 56EC n_0=7 n_1=9 1
LUT_7 8040 n_0=14 n_1=2 0
LUT_8 1000 n_0=15 n_1=1 0
LUT_9 0FF0 n_0=8 n_1=8 –
LUT_{10} CCCD n_0=7 n_1=9 1
LUT_{11} 0220 n_0=14 n_1=2 0
LUT_{12} 88F8 n_0=9 n_1=7 0
LUT_{13} F128 n_0=9 n_1=7 0
LUT_{14} 3F00 n_0=10 n_1=6 0
LUT_{15} 0001 n_0=15 n_1=1 0
LUT_{16} 9B94 n_0=8 n_1=8 –
LUT_{17} AABA n_0=7 n_1=9 1
LUT_{18} A340 n_0=11 n_1=5 0
LUT_{19} B0E1 n_0=9 n_1=7 0
LUT_{20} F0C0 n_0=10 n_1=6 0

Fig. 4 The example of *RS*-vector value calculation

Stage 2. The compression of the obtained *RS*-vector with the help of any lossless compression method is performed. As a result, a compressed vector RS_{com} is formed. E.g. applying the Huffman's algorithm [46] for the considered *RS* vector we obtain the following system of uneven prefix codes: $000 \rightarrow 1$, $001 \rightarrow 00$, $011 \rightarrow 010$, $100 \rightarrow 011$. Substituting the obtained codes in the *RS*-vector we get the 12-bit vector $RS_{com} = 011010001001$. The length differences of the vectors *RS* and RS_{com} is $\Delta = 6$.

Stage 3. The DWM with monitoring information (binary succession, whose length is not more than Δ) is attached to the vector RS_{com}. In Fig. 5 the values of bits of the initial *RS*-vector, compressed vector RS_{com} and the example of attaching the $DWM = 011100$ are shown. As a result of concatenation of the vector RS_{com} and DWM, we obtain the vector RS^* for the resultant IO.

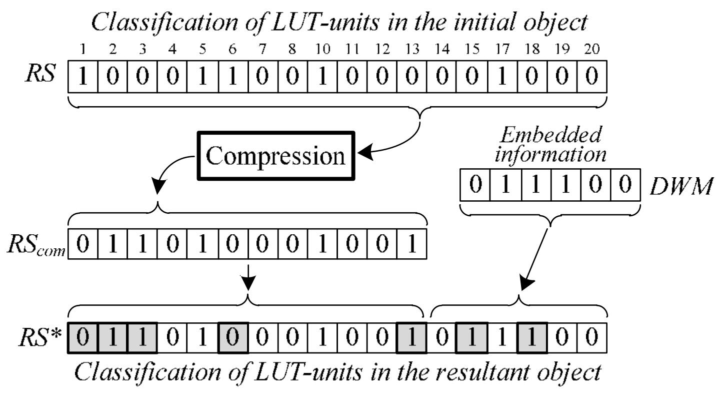


Fig. 5 The binary vector *RS** creation

Stage 4. While examining the embedding path units the *U*-units are ignored, and *R* and *S* units are modified to introduce them in the vector *RS**. In Fig. 5 the values of vector *RS**, which are different from the corresponding values of vector *RS,* are distinguished. The program codes of LUT-units 1, 2, 3, 6, 13, 15, 18 of the initial object are to be inverted in accordance with the principles of equivalent conversions [23] to transfer to the vector *RS** values.

The monitoring information embedded in IO can be obtained by applying the hashing methods, which are differently cryptographically strong and have different amount of bits. Hash sums obtained with the help of various methods can give the different number of LUT-units, which require their code inversion. On this basis before *Stage 3* the authors offer to carry out the process of hashing not with a single method but with the several ones taken from the previously stipulated list in order to choose the best variant of hash sum for the given IO. Similarly the RS vector compression can be performed with some alternative methods to choose the RS_{com} vector variant, which gives the minimum amount of LUT-units requiring their code inversions (Fig. 6). In this case the chosen hashing method for IO and compression ones for RS-vector are to become components of the key to extract the monitoring information out of IO.

The procedure of DWM extraction and the initial form recovery of LUT-object for the suggested method is the following.

Stage 1. The sequential examination of the LUT-object units is performed in the order determined by the embedding path. Wherein the binary vector *RS′* is formed on the basis of the LUT-unit classification.

Stage 2. The last Δ bits, which contain DWM with monitoring information are read from the vector *RS′*.

Stage 3. The rest bits of the vector *RS′* are subjected to the decompression procedure, which is reverse to the compression performed at the stage of the information embedding. As a result the binary vector RS_{decom} is created.

Stage 4. According to the vector RS_{decom} values, the initial IO form recovery is produced. For this purpose, the LUT-unit classes are modified in accordance with the vector RS_{decom} values with the help of equivalent inversion.

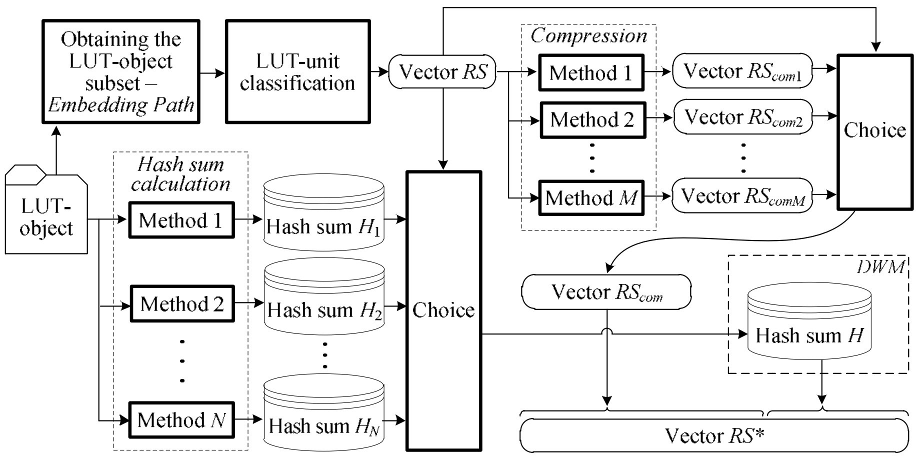


Fig. 6 Search procedure of vector *RS** minimizing the amount of LUT-unit code inversions

In order to confirm the FPGA-project, functioning invariability before and after the monitoring information being embedded in them an experiment was made. The 40 projects for FPGA Altera Cyclone II were used in the experiment. In the course of the project implementation the processing speed changes, energy consumption and thermal dissipation were researched. These characteristic measurements were carried out with the help of modules CAD Altera Quartus II Timing Analyzer and Power Play. The changes of processing speed were 0.18%, energy consumption and thermal dissipation – 0.22%, on the average. As one can see the change values are present within the limits of error value of measurement means. So the results of the experiment have confirmed that LUT-objects, in which the monitoring information was embedded with the help of the proposed method, took their original forms after this information extraction.

6 Conclusions

Development of Green IT Engineering on the way of increase in energy efficiency and safety of FPGA projects is restricted to domination of array structures in circuitry decisions. The main shortcoming of array structures consists in their low checkability creating a problem of the hidden faults in safety-related systems. These faults can be accumulated in matrix circuits in the normal mode and reduce fault tolerance of circuits and safety of systems in the most responsible emergency mode. Increase in a checkability of circuits requires reduction of array structures.

The way of array structure reduction is perspective, but requires overcoming traditions in use of matrix decisions which passed a long way of enhancement and show high rates in throughput and energy efficiency.

We suggest soft and cardinal ways of array structures reduction. The soft way is based on execution of the truncated arithmetical operations implemented in the reduced array structures. The truncated operations are the main method in processing of approximate data – mantissas of floating-point numbers. The cardinal way consists in transition from array structures to bitwise pipelines which parallelize calculations in serial codes.

Experiments on FPGA showed that bitwise pipelines do not interfere with matrix circuits in throughput, but they consume more energy. The suggested method of pipeline multiplication with processing of two bits in a clock cycle allows to increase throughput and to reduce energy consumption.

Development of on-line testing methods in checking of mantissas by inequalities showed increase in trustworthiness of methods for the truncated operations in comparison with residue checking.

Advantage of FPGA design in diversity of a program code is used in the suggested method for an increase in the functional safety of FPGA projects in opposition to the hidden faults. Continuous use of several program codes for the circuits with LUT-oriented architecture allows to exclude accumulation of the hidden faults in a normal mode of safety-related systems.

We propose to use diversity of a program code also in suggested method for monitoring in integrity of the FPGA project in support of security and functional safety of digital components in structure of safe systems.