An Increase in Trustworthiness of Result Checking in Arithmetic Components of Embedded Systems

Oleksandr Drozd[0000-0003-2191-6758], Oleksandr Martynyuk[0000-0003-1461-2000], Kostiantyn Zashcholkin[0000-0003-0427-9005], Julia Drozd[0000-0001-5880-7526]

Odessa National Polytechnic University, Ave. Shevchenko 1, 65044 Odessa, Ukraine
drozd@ukr.net, anmartynyuk@ukr.net, const-z@te.net.ua, dea_lucis@ukr.net

Abstract. The extensive use of embedded systems to process approximate data from sensors in critical, cyber-physical, IoT and other applications orients on-line testing methods to check the validity of approximate results in arithmetic operations. Traditional methods of on-line testing are designed for exact data, i.e. integer by nature. The development of the data model from exact to approximate form changes the purpose of on-line testing for arithmetic components from fault detection to estimation of trustworthiness of the calculated results. The approximate result contains the most and least significant bits, in which the faults of the digital circuit produces essential and inessential errors with respect to the trustworthiness of the result. As a rule, the approximate calculations are characterized by a low probability of an essential error. Traditional on-line testing methods had high error detection probability and low trustworthiness, which approached the probability of an essential error. We propose a method of on-line testing with simplified operation in checking to increase trustworthiness in conditions of low probability of essential error. The method monitors the result of an operation on a limited set of inputs while maintaining the ability to detect typical array circuit faults in the same way as residue checking. The method uses conditions that restrict the input data and logical operations with them. An error detection circuit has been developed and an example of on-line testing of the iterative array multiplier has been considered. The advantages of the suggested method in trustworthiness compared to residue checking are shown.

Keywords: Embedded system, Arithmetical components, On-line testing, Data model, Approximate data processing, Simplified operation in checking, Limiting condition, Logic operation, Trustworthiness.

1 Introduction

On-line testing plays an important role in maintaining the functionality of the embedded system by evaluating the results calculated at the outputs of the digital circuits in its components [1, 2]. The embedded systems are widely used thanks to the success in CAD development, which ensures the rapid design of hardware solutions on program-

mable logic, for example, on FPGA (Field Programmable Gate Array) [3, 4]. They become the basis for the development of components for critical applications, cyber-physical and IoT systems [5–7]. It should be noted that these systems receive initial data from sensors, i.e. results of measurements that relate to approximate data. On-line testing tracks the development of embedded systems and also become focused on the processing of approximate data, which is usually performed in floating-point formats [8, 9].

The main stage of on-line testing development took place within the model of exact data, i.e. integer by nature. This model is reflected in the theory and practice of totally self-checking circuits [10, 11]. According to this theory, the purpose of on-line testing is to detect faults of the digital circuit during basic operations using the first error of the monitored result [12, 13].

The orientation of modern embedded systems and information technologies implemented in them towards the dominance of approximate calculations is not an accident, but, on the contrary, is of a natural property, which is explained from the perspective of the resource approach [14, 15]. This approach, which considers models, methods and means as resources to solve problems, analyzes the process of integrating the human-created computer world into the natural one. The whole history of development of the computer world is evidence of its structuring to the peculiarities of the natural world, among which parallelism and fuzziness have been most evident. This process can be seen in the development of personal computers, which permanently increase the level of hardware support for approximate computing, from the Intel 287/387 coprocessor of optional delivery to several floating-point pipelines in the Pentium family central processor and several thousand such pipelines in the graphic processor. CUDA technology provides for their simultaneous use for execution of parallel calculations [16, 17]. It should be noted that such a natural process of personal computer development has increased productivity from kHz to GHz over 20 years and increased memory from Mb to Tb, that is, the main indicators have improved millions of times at the same time. The progress achieved is attributed to following the development vector.

Increased efficiency of on-line testing is also stimulated by its development along the natural path with increased level of parallelism and fuzziness following the objects of diagnostics. The purpose of this paper is to show the need to improve the data model, transforming it from an exact form to an approximate one for the on-line testing of embedded systems. We suggest to consider the impact that the development of the data model has on on-line testing in its purpose and the trustworthiness of methods. Section 2 deals with the trustworthiness of on-line testing methods in the context of improved data models. The high trustworthiness of traditional methods is found to be low within the approximate data model. Section 3 proposes a method of on-line testing with simplification of operation in checking to increase trustworthiness of monitoring the results calculated by embedded systems. The proposed method is shown on the example of on-line testing for iterative array multiplier.

2 Trustworthiness of on-line testing methods

2.1 Impact of the Data Model on the Trustworthiness of the Methods

Approximate data processing shows the insolvency of the on-line testing purpose declared in the theory of totally self-checking circuits. On-line testing is aimed not at detection of faults in digital circuits, but at estimation of trustworthiness of calculated results. These two goals are indistinguishable within a model of exact data. In this case, the detected error indicates both the fault and the non-reliable result distorted by it [18].

Unlike exact data, the approximate result can be both erroneous and reliable because it consists of most and least significant bits [19]. Circuit faults cause errors in these bits that are respectively essential and inessential to the trustworthiness of the result.

In practice, on-line testing distinguishes between objectives developed within exact and approximate data models. We can see this by detecting a transient fault as a short-term self-eliminating fault. Transient fault causes much more often than permanent one [20, 21]. Therefore, the first result error in totally self-checking circuits is typically caused by a transient fault. The detection of this error is dictated by the desire to assess the trustworthiness of the result. Fault detection is not important because the circuit will be serviceable again after the transient fault.

The exact result contains only most significant bits, in which all errors are essential. The fault detection purpose ignores the difference between essential and inessential errors because least significant bits and inessential errors are not present within the exact data model.

The method of on-line testing is as reliable as it correctly assesses the trustworthiness of the result. Therefore, the trustworthiness of the on-line testing method is determined by the following formula [22, 23]:

$$T = P_E P_D + (1 - P_E)(1 - P_D), \tag{1}$$

where P_E – probability of an essential error;

P_D – probability of error detection.

Formula (1) shows a particular case of exact data where all errors are essential and $P_E = 1$. This is why the trustworthiness of the on-line testing methods is the same for exact data with the probability of error and fault detection, i.e. $T = P_D$.

Traditional solutions using totally self-checking circuits provide fault detection from a given set with $P_D = 1$ probability. In this case, the trustworthiness of traditional on-line testing methods, for example residue checking, is determined by the formula (1) as $T_T = P_E$.

The value of P_E probability can be estimated based on the following judgements. Multiplication is a key operation of approximate calculations because it is used in the representation of numbers in floating-point formats: $(-1)^{\text{SIGN}} \times B^{\text{EXPONENT}} \times \text{SIGNIFICAND}$, where B is the base of the number system [24, 25]. This is why all operations with mantissas contain a multiplication operation or its particular case, and the results of these operations inherit the properties of the product. One of these properties is to double the size of the product compared to the operand for two-operands operations. However, the mantissa of the result must inherit the mantissa size of the

operand. It leads to rejection of a younger half of the calculated result and reduction of P_E probability twice that limits it to the $P_E \leq 0.5$ level. Renormalization and normalization operations performed with operands and results further reduce the P_E probability for the results of all previous and subsequent operations, respectively. Indeed, the renormalization of operands is performed with the alignment of the exponent and the loss of the lower bits in the mantissa of the operand with the smaller exponent. However, these bits were most significant in the results of all previous operations. Errors in ejected bits become inessential. Normalization of the result reduces the number of most significant bits in its representation and in the results of all the following operations, also reducing the P_E probability of an essential error [26, 27].

Thus, the T_T trustworthiness of traditional methods is as low as the P_E probability. Their high P_D probability is mainly used to detect the most frequently occurring inessential errors, i.e. to reject erroneous but reliable results.

2.2 Improving the Trustworthiness of On-Line Testing Methods

Analysis of formula (1) shows that the trustworthiness $T = 0.5$ if at least one of the P_E or P_D parameters takes such a value, i.e. $P_E = 0.5$ or $P_D = 0.5$.

Trustworthiness $T > 0.5$ is achieved if both parameters P_E and P_D are on one side of value 0.5, i.e. in two cases, which determine two ways to increase trustworthiness of on-line testing methods:

1) $P_E > 0.5$ and $P_D > 0.5$;
2) $P_E < 0.5$ and $P_D < 0.5$.

Note that formula (1) is symmetric, i.e. it does not change when the P_E and P_D parameters are changed. However, these parameters play a different role in it: the P_E probability of an essential error characterizes the object of diagnosis, and the P_D probability of error detection – the method of on-line testing. Thus, the task for the on-line testing is determined on the basis of the required trustworthiness T and the characteristic P_E of the diagnostic object. By these parameters, the P_D probability of error detection is determined according to formula (1) as follows:

$$P_D = (T + P_E - 1) / (2P_E - 1), \tag{2}$$

The value of P_D probability obtained in formula (2) is used to select the on-line testing method.

The first way to increase the trustworthiness of on-line testing methods is only possible in the case of $P_E > 0.5$, which is excluded when performing complete arithmetic operations. The truncated operations calculate the result, which is twice or almost twice as short as the complete [28, 29].

For example, the truncated multiplication of n-bit operands defines the $(n + \log_2 n)$-bit product. In this case, the probability of an essential error can be estimated as $P_E = n / (n + \log_2 n)$. The probability is $P_E = 0.86$ and $P_E = 0.91$ for $n = 32$ and $n = 64$, respectively.

The first way is implemented in the residue checking of truncated operations [30, 31]. In addition, the truncated operations can be checked with the limitations imposed

on the normalized numbers by floating-point formats [32, 33]. These constraints are the basis for checking methods by inequalities [34].

The advantage of these methods is increased trustworthiness with a high probability of error detection.

However, even truncated operations do not guarantee a high probability P_E of an essential error due to its halving after each multiplication and during normalization / renormalization operations. For example, if the operation follows X multiplication operations and Y addition operations, each of which shifts the operand to the right by $n / 4$ positions with loss of $n / 4$ least significant bits, the P_E probability may be reduced to a value $0.5^X (0.75)^Y P_E$, i.e. up to 0.38 P_E and 0.14 P_E in the case of $X = Y = 1$ and $X = Y = 2$, respectively.

Thus, the case of $P_E << 0.5$ is most typical for on-line testing of floating-point arithmetic operations just as the second path becomes the main one in improving trustworthiness.

3 Method of on-line testing with simplification of operation in checking

3.1 Basic Provisions of the Method

We offer a method of on-line testing, which increases its trustworthiness along the second path for diagnostic objects with probability $P_E < 0.5$. In this case, the probability of error detection should also be low, i.e. $P_D < 0.5$. Reduction of P_D probability is achieved by execution of checking for operation simplified by its consideration on limited set of input data [35]. For example, multiplication $A \times B$ can be checked as a simpler squaring operation on a set of inputs satisfying the following condition: $A = B$.

The method lowers the P_D probability to a δP_D value, where the reduction coefficient can be estimated as $\delta = H / G$, where H and G are the size of the limited and of the total set of input data, respectively.

The main requirement for the proposed method is to detect errors caused by typical faults of monitored arithmetic units. A set of such faults can be determined based on the capability of the residue checking modulo-three method, i.e. the proposed method should detect errors produced by all faults F that are detected by the modulo-three checking. This method is chosen as a reference in detecting of a set of faults based on our experiments. We have developed a program model of the Brown multiplier [36] with the introduction of a fault of a short circuit between two points in the scheme of a randomly selected operational element. This fault is characteristic of matrix structures [37]. The simulation showed that modulo-three checking detects the first error caused by any such fault. Note that many of these faults contain all stuck-at faults, which are considered to be the closure of circuit points to a level of logical zero or one. The addition schemes show the same effect.

The basis of the proposed method is conditions limiting the number of test words, i.e. input words, on which the result of the operation is monitored. To store the F set of faults, conditions are generated based on the modulo-three casting out operation.

The modulo-three residue takes 4 values: $1 = 01_2$, $2 = 10_2$, $+0 = 00_2$ and $-0 = 11_2$. In the theory of totally self-checking circuits, codes 01_2 and 10_2 are called allowed and codes 00_2 and 11_2 are called forbidden.

Definition. Conditions that restrict the result of an operation equally or differently are called dependent and independent, respectively.

For example, for the $A \times B = V$ multiplication operation, conditions $A \ mod\ 3 = 0$ and $B \ mod\ 3 = 0$ are dependent because they equally limit the complete product: $V \ mod\ 3 = 0$. The product can be presented by the older V_H and the younger V_L half. The older V_H part is a rounded result. The younger V_L part is discarded. Thus, the condition $V_L \ mod\ 3 = 0$ is independent with respect to the first two conditions.

Multiple conditions require you to define the logical operations to perform with them. For example, the dependent conditions considered may be executed simultaneously:

$$(A \ mod\ 3 = 0) \text{ AND } (B \ mod\ 3 = 0)$$

or at least one of them:

$$(A \ mod\ 3 = 0) \text{ OR } (B \ mod\ 3 = 0)$$

or only one of them:

$$(A \ mod\ 3 = 0) \text{ XOR } (B \ mod\ 3 = 0).$$

It should be noted that each of the dependent conditions considered limits the set of test words as $H = G / 3$, which determines the coefficient $\delta = 1 / 3$.

The logical operations AND, OR, and XOR define a coefficient δ to the operations with the sets:

$$\delta_{AND} = 1 / 3 \times 1 / 3 = 1 / 9 \approx 11.1\%;$$
$$\delta_{OR} = 1 / 3 + 1 / 3 - \delta_{AND} = 5 / 9 \approx 55.6\%;$$
$$\delta_{XOR} = 1 / 3 + 1 / 3 - 2\delta_{AND} = 4 / 9 \approx 44.4\%.$$

The logical operations used must retain the F set of faults. Analysis of two-operands logical operations has highlighted two such operations for dependent conditions: OR, XOR, and one operation for independent conditions: AND.

A complete condition that combines logical operations with all dependent and independent conditions used determines a set of all test words. The result constraints define a condition for monitoring it on this set. The reduction coefficient δ of P_D probability is formed by selecting conditions and logical operations thereon.

The minimum value of the δ coefficient may be limited to the allowable fault detection time (number of cycles), which may be estimated as $\tau = ln\ 2 / P_D$. If the probability of error detection decreases compared to residue checking, the reduced probability is the same as the δ coefficient, i.e. the monitoring is performed with the probability $P_D = \delta$.

3.2 Error Detection Circuit

The suggested method monitors the result at the output of the arithmetic unit according to the error detection circuit shown in Fig. 1.

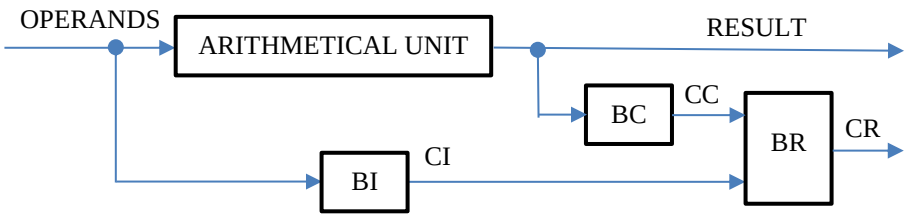

Fig. 1. Circuit of error detection.

The circuit contains a test word identification block BI, a result checking block BC, and a check code generating block BR. The BI block analyzes the operands and generates the *CI* code of inverse identification of a test word. This code takes the forbidden values 00_2 or 11_2 if the operands form a test word and allowed values 01_2 or 10_2 otherwise. The BC block verifies the result according to the condition for its monitoring and generates a result check code *CC* which receives the forbidden values 00_2 or 11_2 in case of condition violation and the allowed values 01_2 or 10_2 otherwise. The BR block receives the *CI* and *CC* codes and generates a *CR* check code that receives the forbidden values 00_2 or 11_2 if both the *CI* and *CC* codes receive the forbidden values 00_2 or 11_2 and the allowed values otherwise.

The BI block allows monitoring of the result by the forbidden values 00_2 or 11_2 so that the fault of the short circuit between the bits of the *CI* code does not cause blocking of the monitoring circuit.

Dependent and independent conditions for operands and condition for monitoring the results are realized using modulo-three casting out unit, which are totally self-checking, i.e. show their own faults.

Logical operations with allowed and forbidden condition codes can be performed in functional-complete basis of operations: AND, NOT or OR, NOT.

The AND logic operation with the allowed codes is performed on a totally self-checking Carter element, which is a modulo-three multiplier [38, 39]. Allowed values 01_2 and 10_2 are not zero. Therefore, the modulo-three multiplication result takes the allowed value, i.e., not equal to zero, in that and only if all the multipliers are not zero and are therefore allowed values.

The OR logic operation with the forbidden values 00_2 or 11_2 is also performed on a modulo-three multiplier. The modulo-three multiplication result takes a forbidden value of zero if at least one of the multipliers is zero, i.e. is a forbidden value.

The NOT logical operation converts allowed codes to forbidden ones and vice versa. The circuit implementing this operation contains one inverter inverting one of the bits of the converted code.

3.3 Monitoring of an Iterative Array Multiplier

The application of the method can be illustrated by an example of an iterative array matrix multiplier monitoring that performs a complete operation with 8-bit operand codes $A\{1, \ldots, 8\}$, $B\{1, \ldots, 8\}$ and calculates the $V\{1, \ldots, 16\}$ product. The old half $V\{1, \ldots, 8\}$ of the product is the result. The P_D probability is set at 9% – 10%.

A method can use two dependent conditions and an OR operation with them:

$$(A\{1, \ldots, 8\} \; mod \; 3 = 0) \text{ OR } (B\{1, \ldots, 8\} \; mod \; 3 = 0).$$

In addition, the lower part of the product $V\{9, \ldots, 16\}$ can be used to create independent conditions, for example:

$$V\{9, 10\} \; mod \; 3 = 0;$$
$$V\{11, \ldots, 16\} \; mod \; 3 = 0.$$

An AND operation is performed with all mutually independent conditions. The condition for result monitoring takes into account the constraints imposed by dependent and independent conditions by the following formula:

$$V\{1, \ldots, 8\} \; mod \; 3 = 0.$$

The error detection circuit is shown in Fig. 2

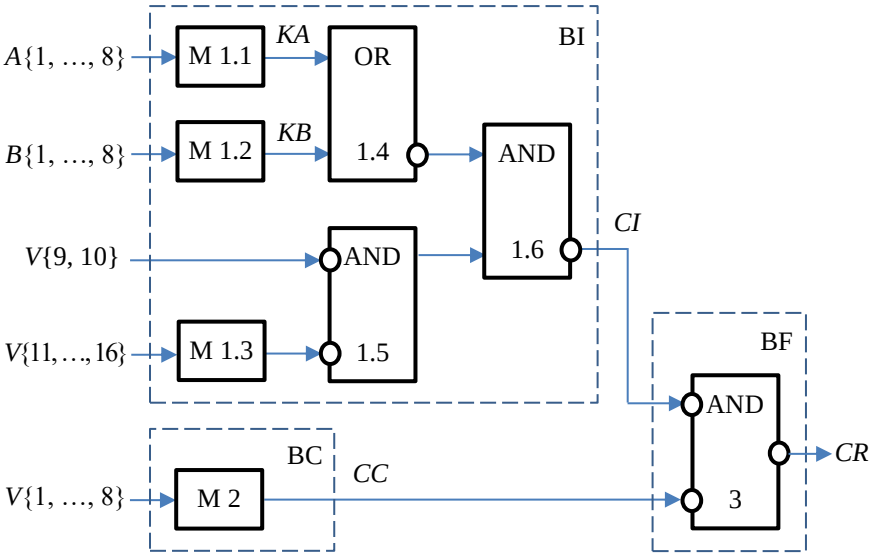

Fig. 2. Circuit of error detection in iterative array multiplier.

The diagram contains modulo-three casting out units M 1.1 - 1.3 and 2, OR unit 1.4, AND units 1.5, 1.6 and 3.

Units M 1.1 and 1.2 calculate dependent condition codes: $KA = A\{1, \ldots, 8\}$ *mod* 3 and $KB = B\{1, \ldots, 8\}$ *mod* 3. The OR 1.4 unit performs an OR operation with the *KA* and *KB* codes as forbidden values and calculates the $K_{A\ OR\ B}$ result in the form of allowed values at the inverse output. The M 1.3 unit calculates the $KV_{11, \ldots, 16} = V\{11, \ldots, 16\}$ *mod* 3 code of the independent condition $V\{11, \ldots, 16\}$ *mod* 3 = 0. The AND 1.5 and 1.6 units carry out the operations AND with $KV_{19, 10} = V\{10, 10\}$ *mod* 3 = $V\{9, 10\}$, $KV_{11, \ldots, 16}$ and $K_{A\ OR\ B}$ codes of independent conditions and calculate the *CI* code at the output of the BI block.

The M 2 unit calculates the $CC = V\{11, \ldots, 16\}$ *mod* 3 code of conditions for the result and transfers it to the output of block BC.

The AND 3 unit receives the *CI* and *CC* codes to the inverse inputs and performs AND operation with the allowed values. The result of the operation is generated at the inverse output of the unit and is transferred to the output of the BF block and the circuit as the *CR* code. Forbidden *CR* code values indicate an error detection on the test word. Allowed values are generated if the correct result is calculated on the check word, or if the input word is not a test word.

The P_D probability is estimated taking into account the probability of performance of the dependent and independent conditions, as well as logical operations with them according to the following formula:

$$P_D = \delta = \text{AND}\ (\text{OR}\ (\delta_{KA}, \delta_{KB}), \delta_{KV\{9, 10\}}, \delta_{KV\{11, \ldots, 16\}}), \qquad (3)$$

где $\delta_{KX} = X_1 / X_2$, X_1 and X_2 – number of multiple values and all code values *KX*;

$\delta_{KA} = \delta_{KB} = 84 / 256$; $\delta_{KV\{9, 10\}} = 2 / 4$; $\delta_{KV\{11, \ldots, 16\}} = 22 / 64$;

$\delta_{\text{OR}} = \text{OR}\ (\delta_{KA}, \delta_{KB}) = \delta_{KA} + \delta_{KB} - \delta_{KA} \times \delta_{KB}$;

$\delta = \text{AND}\ (\delta_{\text{OR}}, \delta_{KV\{9, 10\}}, \delta_{KV\{11, \ldots, 16\}}) = \delta_{\text{OR}} \times \delta_{KV\{9, 10\}} \times \delta_{KV\{11, \ldots, 16\}}$.

Formula (3) determines the $P_D = 9.4\%$ probability. In this case, the trustworthiness of the proposed method is represented by the formula (1) as $T_S = 0.094\ P_E + 0.906\ (1 - P_E)$.

Fig. 3 shows diagrams of the trustworthiness of the proposed method and the traditional solution (on the example of modulo-three residue checking) versus the P_E probability of an essential error.

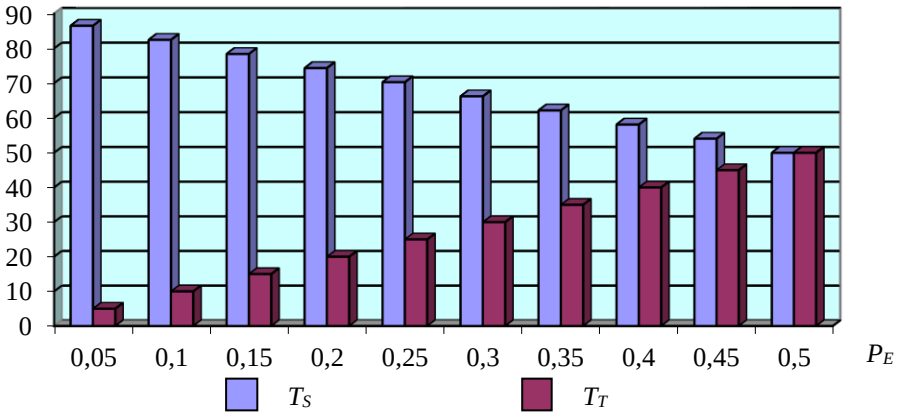

Fig. 3. Trustworthiness of suggested method and residue checking.

Fig. 3 shows the advantage of the proposed method in trustworthiness for the most frequent case of probability $P_E < 0.5$.

We have developed a software model for the error detection circuit that is implemented in the Intel Max 10 FPGA 10M50DAF672I7G [40] using the Quartus Prime 18.1 Lite Edition CAD system [41]. The FPGA project and program model completely confirmed functionality of the error detection circuit.

Note that FPGA design with LUT-oriented architecture (LUT – Look-Up Table) ignores many features of the initial circuit. In particular, the inversion at the input or output of a unit of the error detection circuit may be attributed to a previous or next LUT unit. In this case, fault of short circuit between bits of the *CI*, *CC* or *CR* code causes the circuit to be blocked when the test words and errors are ignored. To eliminate such locks, we fix the inversion position by introducing a branching point.

4 Conclusion

Embedded systems are most commonly used to handle approximate sensor data in critical applications, cyber-physical structures, and IoT solutions. Under these conditions, the on-line testing should also be oriented to approximate calculations.

Traditional on-line testing methods developed within the exact data model lose effectiveness in the trustworthiness of approximate result monitoring.

The proposed method of on-line testing with simplification of check operation is developed for the most frequent case of low probability of essential error in approximate data processing.

The method performs monitoring the operation result on a limited set of input data without reducing the set of faults detected by the traditional residue checking method. The proposed method shows an advantage in trustworthiness over residue checking. This advantage increases with reduced probability of essential error.

References

1. Nicolaidis, M., Zorian, Y.: On-Line Testing for VLSI – a Compendium of Approaches. Electronic Testing: Theory and Application (JETTA). Vol. 12, 7–20 (1998)
2. Coppad, D., Sokolov, D., Bystrov, A., Yakovlev, A.: Online Testing by Protocol Decomposition. In: Proc. of 12th IEEE International On-Line Testing Symposium, pp. 263–268, Como, Italy (2006).
3. Intel FPGA Integer Arithmetic IP Cores User Guide, [Online]. Available: https://www.intel.com/content/www/us/en/programmable/docum entation/sam1395330298052.html
4. Palagin, A., Opanasenko, V.: The implementation of extended arithmetic's on FPGA-based structures. In: Proc. of the 9th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications, IDAACS'2017, vol. 2, pp. 1014–1019, Bucharest, Romania (2017.
5. Drozd, O., Antoniuk, V., Nikul, V., Drozd, M.: Hidden faults in FPGA-built digital components of safety-related systems. In Proc. of the 14th International Conference "TCSET'2018 Conference "Modern problems of radio engineering, telecommunications and computer science, pp. 805–809, Lviv-Slavsko, Ukraine (2018). DOI: 10.1109/TCSET.2018.8336320

6. Hahanov, V., Litvinova, E., Chumachenko, S.: Cyber Physical Computing for IoT-driven Services. Springer International Publishing, (2017).
7. Kondratenko, Y., Kondratenko, G., Sidenko, I. Multi-criteria decision making for selecting a rational IoT platform. Proceedings of 2018 IEEE 9th International Conference on Dependable Systems, Services and Technologies, DESSERT 2018, pp. 147–152, Kiev, Ukraine (2018).
8. Synopsys, "DWFC Flexible Floating Point Overview," no. August, pp. 1–6 (2016)
9. Xilinx, "LogiCORE IP floating-point operator v7.0, product guide, PG060," www.xilinx.com/support/documentation, 2014.
10. Anderson, D. A., Metze, G.: Design of Totally Self-Checking Circuits for n-out-of-m Codes. IEEE Trans. on Computers, vol. C-22, 263–269 (1973).
11. Diaz, M., Azema, P., Ayache, J. M.: Unified Design of Self-Checking and Fail-Safe Combinational Circuits and Sequential Machines. IEEE Trans. Computers, vol. C-28, 276–281, (1979).
12. Metra, C., Schiano, L., Favalli, M., Ricco, B.: SelfChecking scheme for the on-line testing of power supply noise. In: Proc. of Design, Automation and Test in Europe Conference, pp. 832–836, Paris, France (2002).
13. Kubalik, P., Fisher, P., Kubatova, H.: Fault tolerant system design method based on self-checking circuits. In: Proc. 12th IEEE International On-Line Testing Symposium, pp. 185–186, Como, Italy (2006).
14. Drozd, J., Drozd, A., Antoshchuk, S.: Green IT engineering in the view of resource-based approach. In: Kharchenko, V., Kondratenko, Y., Kacprzyk, J. (eds.) Green IT Engineering: Concepts, Models, Complex Systems Architectures, Studies in Systems, Decision and Control, vol. 74, pp. 43–65. Springer International Publishing, Heidelberg (2017). doi: 10.1007/978-3-319-44162-7_3
15. Drozd, O., Kharchenko, V., Rucinski, A., Kochanski, T., Garbos, R., Maevsky, D.: Development of Models in Resilient Computing. In: Proc. of 9th IEEE International Conference on Dependable Systems, Services and Technologies (DESSERT'2019), pp. 2–7, Leeds, UK (2019). DOI: 10.1109/DESSERT.2019.8770035
16. Andrecut, M.: Parallel GPU implementation of iterative PCA algorithms. Journal of Computational Biology, vol. 16, no. 11, pp. 1593–1599 (2009). Online. [Available]: http://dx.doi.org/10.1089/cmb.2008.0221
17. NVIDIA CUDA Compute Unified Device Architecture. Programming Guide / Version 1.0, NVIDIA Corporation (2007).
18. Drozd, J., Drozd, A., Al-dhabi, M.: A resource approach to on-line testing of computing circuits. In: Proc. IEEE East-West Design & Test Symposium, pp. 276 – 281, Batumi, Georgia (2015). DOI: 10.1109/EWDTS.2015.7493122
19. Kekre, H.B., Mishra, D., Khanna, R., Khanna, S., Hussaini A.: Comparison between the basic LSB Replacement Technique & Increased Capacity of Information Hiding in LSB"s Method for Images. International Journal of Computer Applications. 45, No. 1, 33–38 (2012).
20. Pizza, M., Strigini, L., Bondavalli, A., Di Giandomenico, F. Optimal discrimination between transient and permanent faults. In: 3rd IEEE High Assurance System Engineering Symposium, pp. 214–223, Bethesda, MD, USA (1998).
21. Favalli, M., Metra, S.: Problems due to Open Faults in the Interconnections of Self-Checking Data Path. In: Proc. of IEEE Design, Aut. and Test in Europe, pp. 612–617, Paris, France (2002).
22. Drozd, A., Lobachev, M., Drozd, J.: The problem of on-line testing methods in approximate data processing. In: Proc. of 12th IEEE International On-Line Testing Symposium, 251–256, Como, Italy (2006). DOI:10.1109/IOLTS.2006.61

23. Drozd, A., Antoshchuk, S.: New on-line testing methods for approximate data processing in the computing circuits. In: 6th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications, pp. 291–294, Prague, Czech Republic (2011). DOI:10.1109/idaacs.2011.6072759
24. ANSI/IEEE Std 754-1985. IEEE Standard for Binary Floating-Point Arithmetic (1985)
25. IEEE Std 754™-2008 (Revision of IEEE Std 754-1985) IEEE Standard for Floating-Point Arithmetic. IEEE 3 Park Avenue New York, NY 10016-5997, USA (2008)
26. Castillo, A.: Design of single precision float adder (32-bit numbers) according to IEEE 754 Standard using VHDL. In: Master Thesis. Slovenská Technická Univerzita, Bratislava, Slovak Republic, (2012).
27. Ehliar, A.: Area efficient floating-point adder and multiplier with IEEE754 compatible semantics. In: Proc. of International Conference on Field-Programmable Technology, pp. 131–138 (2014).
28. Park, H.: Truncated Multiplications and Divisions for the Negative Two's Complement Number System. In: Ph.D. Dissertation. The University of Texas at Austin, Austin, USA (2007)
29. Garofalo, V.: Truncated Binary Multipliers with Minimum Mean Square Error: Analytical Characterization, Circuit Implementation and Applications. In: Ph.D. Dissertation, University of Studies of Naples "Federico II", Naples, Italy (2008)
30. Drozd, A.V., Lobachev, M.V., Hassonah, W.: Hardware check of Arithmetic Devices with Abridged Execution of Operations. In: the European Design & Test Conference (ED & TC 96), p. 611, Paris, France (1996).
31. Drozd, A., Lobachev, M.: Efficient On-line Testing Method for Floating-Point Adder. In: Proc. of IEEE Design, Aut. and Test in Europe (DATE), pp. 307–311, Munich, Germany (2001). DOI: 10.1109/DATE.2001.915042
32. Langhammer, M.: Floating point datapath synthesis for FPGAs. In: Proceedings - 2008 International Conference on Field Programmable Logic and Applications, FPL, pp. 355–360 (2008).
33. Altera, Ug01058-6.0: Floating-point megafunctions (2011). [Online]. Available: http://www.altera.com/literature/ug/ug altfp mfug.pdf
34. Drozd, A., Antoshchuk, S., Drozd, J., Zashcholkin, K., Drozd, M., Kuznietsov, N., Al-Dhabi, M., Nikul, V.: Checkable FPGA Design: Energy Consumption, Throughput and Trustworthiness. In: book: Green IT Engineering: Social, Business and Industrial Applications, Studies in Systems, Decision and Control, Kharchenko, V., Kondratenko, Y., Kacprzyk, J. (eds), vol. 171, pp. 73–94. Berlin, Heidelberg: Springer International Publishing (2018). DOI: 10.1007/978-3-030-00253-4_4.
35. Chernov, S., Titov, S., Chernova, L., Gogunskii, V., Chernova, L., Kolesnikova, K.: Algorithm for the simplification of solution to discrete optimization problems. Eastern-European Journal of Enterprise Technologies 3 (4), 1–12 (2018).
36. Nicolaidis, M., Manich, S., Figueras, J.: Achieving Fault Secureness in Parity Prediction Arithmetic Operators: General Conditions and Implementations. In: Proc. European Design and Test Conference, pp. 186–193, Paris, France (1996).
37. Pleskacz ,W., Jenihhin, M., Raik, J., Rakowski, M., Ubar, R., Kuzmicz, W.: Hierarchical Analysis of Short Defects between Metal Lines in CMOS IC. In: 11th Euromicro Conference on Digital System Design Architectures, Methods and Tools, pp. 729–734, Parma, Italy (2008).
38. Carter, W., Schneider, P.: Design of Dynamically Checked Computers. In: Proc. IFIP Congress 68, pp. 878–883, Edinburgh, Scotland (1968).
39. Sellers, F. F., Hsiao, M. J., Beamson, L. W.: Error detecting logic for digital computers. McGraw-Hill, New York (1968).
40. Max 10 FPGA Device Architecture (2017), https://www.intel.com /content/dam/www/programmable/us/en/pdfs/literature/hb/max-10 /m10_architecture.pdf

41. Intel Quartus Prime Standard Edition User Guide: Getting Started. Available at: https://www.intel.com/content/dam/www/programmable/us/en/pdfs/literature/ug/ug-qps-getting-started.pdf