Component-Based Safety-Oriented On-Line Testing of Digital Systems

A. Drozd [1], V. Kharchenko [2,3], A. Siora [4], V. Sklyar [2,3]
[1]*Odessa National Technical University*
[2]*National Aerospace University named after N.E. Zhukovsky "KhAI"*
[3]*Centre for Safety Infrastructure-Oriented Research and Analysis*
[4]*Research and Production Corporation "Radiy"*
[1]*drozd@ukr.net*
[2,3]*V.Kharchenko@khai.edu*
[4]*Marketing@radiy.com*

Abstract

Principles and methods of on-line testing for digital components and systems are analysed. The features of diagnostics for safety-critical NPP I&C systems are described. The paper zeroed in on component-based approach to safety-oriented on-line testing. Several methods, techniques and schemes for on-line testing of digital components of safety critical I&C systems are proposed. These methods are based on using of natural information redundancy and reduced checking of the components.

1. Introduction

1.1. Motivation

Safety of critical applications, for example, NPP Instrumentation and Control systems (I&C systems), on-board aerospace systems, etc. directly depends on reliability and availability of digital devices and systems. There are non-functional requirements to safety and dependability of hardware and digital I&C systems described in the international and national standards including requirements to diagnostics trustworthiness [1].

The most rigorous control functions of safety-critical NPP I&C systems (results of their performing) should be checked by on-line testing means in real-time [2]. Since the results of such testing are codewords, then the correct result, for which every error caused by a circuit fault is significant (that is error that leads to incorrect result), is reliable result. In this case, validity checking of the result that aimed at detection of significant error coincides with a problem of circuit fault detection, which is stated in self-checking circuit theory allowing us to use its groundwork [3].

1.2. Analysis of the related works. Main conceptions

In according to self-checking circuit theory, every circuit is completely self-checking in specified fault domain if it is protected and self-testing in this fault domain [3-5].

A circuit is fault-safe for a set of faults F if for every fault in F the circuit never produces an incorrect codeword at the output for an input codeword.

A circuit is self-testing for a set of faults F if for every fault in F the circuit produces a non-codeword at the output for at least an input codeword.

If the circuit is both fault-safe and self-testing it is said to be totally self-checking.

Protected circuit is interpreted subject to a result of information redundancy of the code, i.e. decomposition of result's codeword set into correct codewords, which are generated during correct operation of the circuit, and incorrect codewords, which are generated during error. A set of incorrect codewords is not empty. Moreover, it is important to mark a set of input codewords that used by a system during its correct operation. These codewords are treated as operational codewords. Then the system is fault-safe for a specified set of faults, if for every fault the circuit produces correct or incorrect codeword at the output for an input operational codeword.

A circuit is self-testing for the specified set of faults if for every fault the circuit produces incorrect output codeword for at least a single operational input codeword.

If the fault causes error for t bits and $t < d$, where X is Hamming code distance between the data and the rest codewords, then the circuit is protected since it computes incorrect output codeword. Code distance is provided by introducing of check bits. Therefore, cir-

cuit security defines information redundancy required to detect single fault via single error.

Circuit self-testability is aimed at allowing of single fault detecting before next fault. The condition assumes that time interval between such faults contains all possible operational input codewords. Such situation is possible when the faults are rare and when a system operates at high frequency, i.e. when circuit design is highly-reliable and productive.

Development of self-testing circuits allows to detect faults of primary and checking equipment using operational input codewords [6-9].

Considering the principles of I&C systems development, i.e. development using library components, on-line testing of I&C system can be developed along component on-line testing (COT), where tests lie at the level of components in a form of solutions to develop them during I&C system integration process.

1.3. Goal and structure

It is necessary to solve the following problems to implement COT:

– development of checking circuit (CKK) for each component (component checking circuit generates two-digit checking code K with one of two possible values: 01 or 10, when both the component and checking circuit operate correctly, and value of 00 or 11 when error is detected);

– integration of CKK into I&C system;

– development of processing circuit for checking codes (COK) K received from separate components.

These problems will be reviewed in Sections 2, 3, and 4, respectively. Section 5 states some solutions concerning checking of I&C components.

2. Development of checking circuit for a component

Single-output components are typically used in I&C systems.

Single-output component can be checked using duplication via main component (OK) complementation by a checking component (KK). Duplication can also be the best solution for multi-output components (control circuits with or without memory); such solution is the simplest and can provide the largest test coverage for the results.

KK implements inversion function for the result. In this case both the result and independently computed inversion form the checking code.

One can propose two basic ways of KK development at duplication level:

– as a copy of OK with inverse output, i.e. follow the description of OK;

– as a solution, which is dual to OK.

3. Integration of CKK into I&C system

One can propose two basic connection diagrams for integration of CKK into I&C system:

– diagram that receives input data for KK from precede OK outputs (Fig.1, a);

– diagram that receives input data for KK from precede KK outputs (Fig.1, b).

Figure 1. Basic connection diagrams for integration of CKK into I&C system

Disadvantage of the first diagram is that it contains independent inputs for OK and KK. Input fault can be the common fault for both OK and KK, and it will not be detected. Moreover, such fault at input of KK can impact OK operation, i.e. COT negatively impacts on primary equipment of I&C system. Advantage of this diagram is in independence of checking codes K, i.e. each of them reflects the status of the corresponding pair of OK and KK components.

Second diagram consists of main and checking systems composed by OK and KK; it minimizes dependence of inputs (the dependence remains for I&C system inputs only, and there is no dependence in components interconnections). The advantage strengthens as ratio between component interconnections and total number of connections (or input connection of I&C system) increases, i.e. when I&C system has a less degree of parallelization. If all the components are connected to inputs of I&C system in a parallel way, then the advantage disappears. Disadvantages of the second diagram are in dependence with checking codes, i.e. mismatch in operation of main and checking systems will be registered by the series of sequential components: each fault will be registered by several checking codes K that leads to more complicated fault identification. If all the components are connected to

inputs of I&C system in a parallel way, then the disadvantage disappears.

Thus, first and second diagrams coincide with each other when all the components are connected to inputs of I&C system in a parallel way, and they are different in disadvantages that consist of dependence of inputs (or checking codes), which strengthen as number of component interconnections increases.

4. Processing of checking codes

Processing of checking codes, received from the components, is a separate problem. Complexity of such problem increases with the number if I&C system components.

The problem can be stated in several ways:

– obtaining of common checking code that indicates a fault of I&C system in a whole;

– I&C system fault identification, up to a component or a set of components.

In practice, the most used is obtaining of common checking code and fault identification up to a set of components, which forms card module important for patching or reconfiguration.

4.1. I&C system common checking code

Common checking code K_O of I&C system or card module (together I&C system checkpoints) is implemented using well-known (from self-checking circuit theory) compression circuit [5] (Fig. 2, a) with application of Carter elements (chips) (EK – Fig. 2, b) [6].

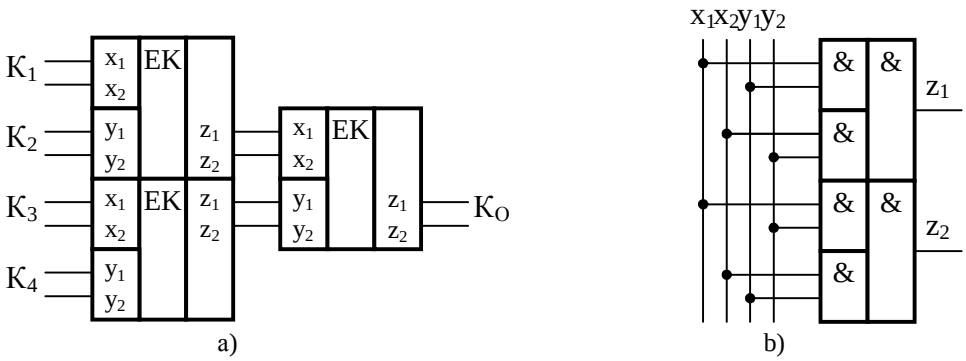

Figure 2. Compression circuits

Carter element integrates two pairs of inter-inverse bits of x_1, x_2 and y_1, y_2 (where $x_1=\neg x_2$ and $y_1=\neg y_2$) into one pair of z_1, z_2, where $z_1=\neg z_2$, using the following expressions:

$$z_1 = x_1 \wedge y_1 \vee x_2 \wedge y_2$$

$$z_2 = x_1 \wedge y_2 \vee x_2 \wedge y_1$$

When at least one pair contains the same bits (i.e. when error detected), the output bits of compression circuit are also the same.

Carter element also detects own constant single faults.

Before self-checking circuit theory, Carter element was known as modulo 3 multiplier. Actually, this element implements modulo 3 multiplication of modulo 3 remainders. Accepted values for modulo 3 remainders are +0, -0, 1, 2, and they are represented by binary codes of 00_2, 11_2, 01_2, 10_2, respectively. If all the multipliers are nonzero, i.e. 01_2 or 10_2, then the product is nonzero. If at least one multiplier is zero (+0, -0), i.e. 00_2 or 11_2, then the product is zero. If checking codes of K_1 and K_2 are compressed by Carter element, then such operation can be described using the following expression: $(K_1 \cdot K_5) \bmod 3$.

If the processing is pipelined, then checking code processing is also being pipelined, i.e. being partitioned by registers. Each partition contains single Carter element or pyramidal compressing circuit of Carter elements; it depends on cycle duration and Carter element delay. Thus, common checking code can be computed after few cycles from receiving of component checking code that is the disadvantage of COK.

4.2. Fault identification of the I&C systems

I&C system fault identification can be implemented in two basic ways:

– indicate all the checkpoints (checking codes or common checking codes for card modules);

– obtain a number of fault component or card module.

First way is more appropriated when number of the components (or card modules) is small. In this case, indication of each (common) checking code requires two modulo 2 adders, two triggers, and two LEDs (main and duplicated).

Modulo 2 adders detect the error via coincidence of

checking code bits and then set their trigger to one (initially set to zero), fixing the error. Error detection and fixation unit consists of two adders and two triggers.

Second way can reasonable be implemented in a parallel and consecutive way for the first and second circuits (OK and KK components connections), respectively.

Proposed parallel implementation is in encoding of checking codes. It can be done to indicate number of fault component in binary, decimal, or other notation. To maximally utilize indication bits, we can recommend using hexadecimal notation (two bits allow to indicate one from 256 component numbers).

Parallel implementation is based on independence of checking codes and can be implemented using the following technique.

1) Checkpoints (components or card modules) being numerated from zero.

2) Checking codes being calculated for every bit of checkpoint number binary code; it can be done by integration of checking codes with the numbers equal to one in this bit.

3) Obtained common checking codes allow to detect and fix the error (using error detection and fixation circuits for every common checking code).

4) Trigger outputs control the indication segments that display a number of fault component or card module.

For example, let I&C system contains 8 checkpoints with $K_0 - K_7$ checking codes numerated from 000_2 to 111_2. Then three common checking codes can be calculated for the first (low-order), second and third bits of the checkpoint binary number, respectively. First common checking code integrates K_1, K_3, K_5, and K_7 checking codes of odd checkpoints that contain one in first bit of the number. Second common checking code integrates K_2, K_3, K_6, and K_7 checking codes of checkpoints that contain one in second bit of the number. Third common checking code integrates K_4, K_5, K_6, and K_7 checking codes of checkpoints that contain one in third bit of the number.

Consecutive implementation can be used for dependent checking codes in component series and consist of sequential scanning of checking codes in according to the order of the components (checkpoints) in series. It can be implemented using two multiplexers (one multiplexer per checking code bit) and counters. Checking code bits arrive at data inputs of the multiplexers, and then, under counter control, sequentially being switched to outputs. Thus, the sequence of checking codes is formed by multiplexer outputs, and then error detection and fixation module generates main and duplicated signals that stop the counters at the number of fault component in a case of error. These numbers being sent to the duplicated indication then.

5. Methods of I&C system component safety-oriented on-line testing

5.1. Using of natural information redundancy

1. Information redundancy in a form of incorrect codewords of binary code can be implemented by introduction of additional checking bits into the code or it can be initially presented in the code. The last case is known as natural information redundancy (NIR) [6].

Detection of NIR is possible by generation of multibit code based on outputs of a single or several components.

To detect NIR it is necessary to form a set of correct codewords for multibit code, count them number r, and compare it with the code capacity n. If $r < 2^n$, then there are $2^n - r$ incorrect codewords; it means that NIR is present.

2. Using of NIR is reasonable when incorrect codeword detection circuit is simple (as well as the procedure of circuit implementation) that can occur in linear code. Therefore, at the next stage, an initial multibit code should be transformed into linear code by obtaining of linear-independent codewords (with respect to modulo 2 sum).

For example, let $n = 5$ and $r = 5$ for multibit code (see Table 1).

Table 1. Multibit code

Codeword \ Bit	1	2	3	4	5
1	1	0	1	1	0
2	0	1	1	1	1
3	1	0	0	1	1
4	0	0	1	0	1
5	1	1	0	0	1

To extract linear-independent codewords, we need to form linear code generating matrix (see Table 2).

Table 2. Linear code generating matrix

Codeword \ Bit	1	2	3	4	5
1	1	0	0	1	1
2	0	1	0	1	0
3	0	0	1	0	1

Generating matrix consists of data bits (1, 2, and 3), which can be described by unit diagonal, and checking bits (4 and 5), for which we can define expressions of linear syndrome, as modulo 2 sum for two data bits:

$$4 = 1 \oplus 2;$$
$$5 = 1 \oplus 3.$$

When all five component outputs are independently implemented, it is possible to identify the fault up to one bit (output function 1) if conditions of both expressions are not hold; it is possible to identify the fault up to two bits (2 and 4, or 3 and 5) if condition of one expression (first or second, respectively) is not hold.

To obtain checking codes for three described sets of outputs (1; 2 and 4; 3 and 5) it is necessary to calculate the following bits: $4^* = \neg(1 \oplus 2)$ and $5^* = \neg(1 \oplus 3)$; during correct operation of components, these bits form with 4 and 5 bits inter-inverse pairs of 4, 4* and 5, 5*, i.e. checking codes (K_4 and K_5).

Then checking codes for sets of outputs (1; 2 and 4; 3 and 5) can be defined using Carter elements and modulo 3 multiplication:

$$K_1 = \neg((\neg K_4 \cdot \neg K_5) \bmod 3);$$
$$K_{2and4} = \neg((\neg K_4 \cdot K_5) \bmod 3);$$
$$K_{3and5} = \neg((K_4 \cdot \neg K_5) \bmod 3),$$

where symbol of inversion «¬» means inverting of checking code single bit.

Thus, the components are under checking without their duplication, due to disclosed NIR of their outputs code only.

5.2. Reduced checking of the components

Reduced checking of the components assumes acceptable reduction of error detection probability in according to required validity of result checking when checking circuit is simpler compared to duplication.

The basis of reduced checking is its construction features.

1. For example, components that based on fixed modulo counters (frequency divider on a constant, signal delay on fixed number of cycles) can be checked using a signature; such signature is formed based on values of counter outputs during counting the constant. Reducing of checking validity is in inverse dependence with the signature capacity, and equipment costs – in direct dependence. Let w be a signature capacity. Then the probability, that the error is missed due to coincidence of the signatures for both correct and incorrect counter operation, can be estimated as 2^{-w}; validity of result checking can be estimated using the following expression:

$$D = 1 - 2^{-w}. \quad (1)$$

Checking costs δ times low against duplication costs, where

$$\delta = w_0 / w. \quad (2)$$

For example, if $w_0 = 16$ and $w = 4$, 6 or 8 validity goes down to $D = 0.938$, $D = 0.984$ or $D = 0.996$, and costs are decreased in $\delta = 4$, $\delta = 2.7$ or $\delta = 2$ times, respectively. An error is detected after counter expired.

2. Another example is provided by comparator-based components, which can be checked in comparison with operation of reduced checking comparators that process high-order bits only. Outputs of main and checking comparators then being compared. If input data are different in high-order bits then error signal being generated. Otherwise the checking being blocked and low-order bits comparison errors will not be detected.

Let w_0,d w – number of bits for the main and reduced comparators, respectively. Then, with $w_0 = 8$ and $w = 4$, functional checking circuit is the following (see Fig. 3).

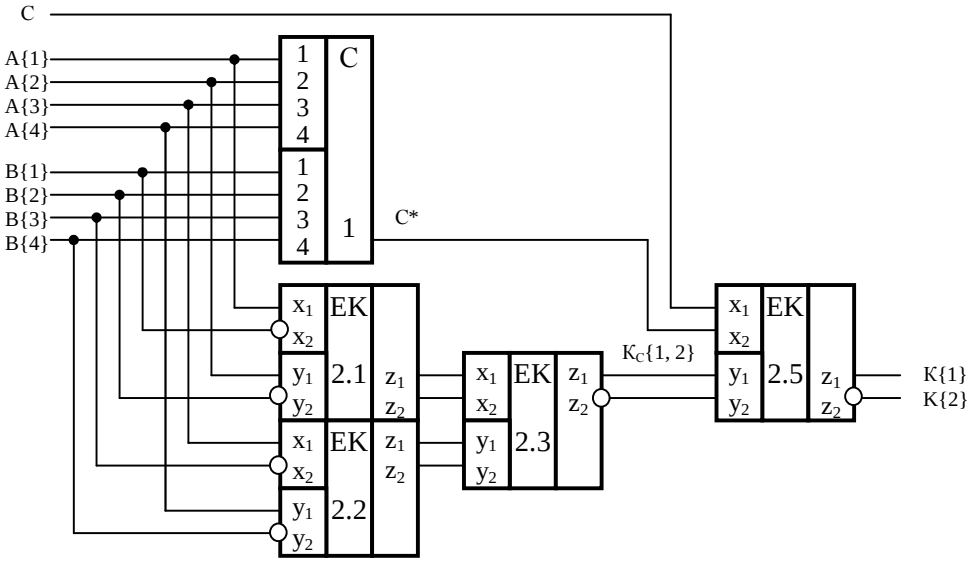

Fig. 3. Comparator checking circuit

Comparator checking circuit contains comparator (C 1) and Carter elements (2.1 ÷ 2.4). Reduced comparator receives high-order bits (A{1} ÷ A{4} and B{1} ÷ B{4}) of compared data (A{1 ÷ 8} and B{1 ÷ 8}). Checking comparator returns comparison bit (C*). High-order bits moves also to the inputs of compression circuit (Carter elements (2.1 ÷ 2.3)), which generates such $K_C\{1, 2\}$ code that $K_C\{1\} = K_C\{2\}$ when high-order bits coincide, and $K_C\{1\} = \neg K_C\{2\}$ otherwise.

Carter element (2.5) receives $K_C\{1, 2\}$ code and a code that consisted of C and C* bits of both main and checking comparators to generate checking code (K) with the value of exact error (the same values of the bits $K\{1\} = \neg K\{2\}$) only when $C \neq C^*$ and $K_C\{1\} = \neg K_C\{2\}$ (i.e. outputs of the comparators do not coincide with high-order bits of compared data).

The probability of missed error can be estimated by the probability of checking blocking (it happens in single case from 2^w), and equipment costs (ignoring Carter elements in checking) are proportional to capacity of the comparator. Then, to estimate validity of result checking and benefits in equipment, we can use expressions (1) and (2), respectively.

A disadvantage of this solution is in absence of operation control for low-order comparator bits; it can lead to accumulation of faults and errors in this part of I&C system.

Conclusion

This paper reviewed some principles and techniques for component-based safety-oriented on-line testing. Component-based on-line testing is based on pre-developed extendible set of checking circuits for both library and designed components, as well as on formalized solutions concerning processing of checking results.

The directions for further research are the following:

1) analysis of complexity restrictions for COT; subject to such restrictions it can be stated and solved a problem of assuring maximal trustworthiness (product of test coverage and error detection probability) under specified additional coverage restriction; the coverage should be equal to one);

2) development of techniques for COT control flag processing (possible variant is time multiplexing), multicomponent construction of OT;

3) research possibilities to develop COT using VHDL code.

5. References

[1] M.A. Yastrebenetsky (edit.) *NPP I&Cs: Problems of Safety*, Ukraine, Kyiv: Technika, 2004. – 472 p. (translated in USA by NPC, 2007)

[2] V.S. Kharchenko, V.V. Sklyar (edits). *FPGA-based NPP I&C Systems: Development and Safety Assessment*, RPC Radiy, National Aerospace University "KhAI", SSTC on Nuclear and Radiation Safety, 2008. – 188 p.

[3] V. Saposhnikov, M. Dmitriev, M. Goessel et al. *Self-Dual Parity Checking – a New Method for on-Line Testing* / // Proc. IEEE VLSI Test Symposium. – 1996. – P. 162 – 168.

[4] M. Nicolaidis, Y. Zorian. *On-Line Testing for VLSI – a Compendium of Approaches* // Electronic Testing: Theory and Application (JETTA). – 1998. – V. 12. – P. 7 – 20.

[5] C. Metra , L. Schiano, M. Favalli, B. Ricco. Self-*Checking Scheme for the on-Line Testing of Power Supply Noise* // Proc. Design, Automation and Test in Europe Conf. Paris (France). – 2002. – P. 832 – 836.

[6] D. A. Anderson, G. Metze. *Design of Totally Self-Checking Check Circuits for m-out-of-n Codes* // IEEE Trans. Comput. – 1977. – V. C – 22, N 3. – P. 263 – 269.

[7] C. Metra, M. Favalli, B. Ricco. *Concurrent Checking of Clock Signal Correctness* // Proc. IEEE Design & Test Conf. – 1998. – P. 42 – 48.

[8] E.S. Sogomonyan, E.V. Slabakov. *Self-Testing Units and Fault-Tolerant Systems*. – Moscow: Radio andCommunication. – 1989. – 208 p.

[9] W. Carter, P. Schneider. *Design of Dynamically Checked Computers* // Proc. IFIP Congress 68. – Edinburgh (Scotland). – 1968. – P. 878 – 883.