ПРОЕКТИРОВАНИЕ И ДИАГНОСТИКА ВЫЧИСЛИТЕЛЬНЫХ СИСТЕМ

УДК 681.5.09
DOI: 10.17223/19988605/41/8

A.Yu. Matrosova, S.A. Ostanin, I.E. Kirienko, E.A. Nikolaeva

A FAULT-TOLERANT SEQUENTIAL CIRCUIT DESIGN FOR STUCK-AT FAULTS AND PATH DELAY FAULTS

The reported study was supported by Russian Science Foundation, research project № 14-19-00218.

This paper presents a fault-tolerant synchronous sequential circuit design based on self-checking system with low overhead. The scheme has only one self-checking sequential circuit, normal (unprotected) sequential circuit and not self-testing checker. It is proved the reliability properties of the suggested scheme both for single stuck-at faults (SAFs) at gate poles and path delay faults (PDFs), transient and intermittent. It is supposed that each next fault appears when a previous one has disappeared. Estimations of the schemes complexity are discussed.
Keywords: fault-tolerant scheme; sequential circuit; self-checking circuit; stuck-at fault; path delay fault; soft error.

In modern military, space, medical, etc. computer systems requirements for hardware reliability are increased. Continuous improvements in CMOS technology entering the nanometer scale has resulted into quantum mechanical effects creating many technological challenges for further scaling of CMOS devices. Nano-scale devices are limited by higher defect rates and increased susceptibility to soft errors (transient or intermittent).

High performance integrated circuits have to be protected not only for single stuck-at faults (SAFs) (transient or intermittent) at gate poles but also for delays that arise in a circuit operation. Delays may be caused by a high level of circuit integration, low voltages and high frequency operation. One of the most widespread and useful in practice delay models is a model of a path delay fault (PDF). In this model, it is considered that for small delays in path elements and connections between its elements a delay in propagating a change in a signal value may exceed an admissible level for a circuit as a whole. This leads to incorrect operation of an entire circuit.

One of the approaches to increase reliability of the system is fault tolerance. A fault-tolerant system is one that can continue the correct performance of its specified tasks in the presence of faults. Fault tolerance is assumed to add some of the redundancy: hardware redundancy, software redundancy, information redundancy or time redundancy.

One of the most common techniques providing the fault-tolerant property is triple modular redundancy (TMR) [1–4]. The basic idea of TMR is to triplicate the circuit and perform a majority vote to determine the output of the system. The main difficulties with TMR are the voter (if the voter fails, the complete system fails) and high area overhead.

In the paper [5] a finite state machine (FSM) based fault tolerance technique for sequential circuits is proposed. The technique is connected with adding redundant equivalent states to protect states with high probability of occurrence. The added states guarantee that all single faults appearing in the states variables of highly occurring states or in their combinational logic are tolerated.

In [6] the use of duplication with self-checking for one-hot encoding FSM is suggested. This approach ensures that any single error will not lead to an incorrect next state. However, it might lead to an erroneous output. The area overhead of this approach is large as two sets of selectors are used and each selector set has N flip-flops, where N is the number of states.

In the work [7] the synthesis of totally self-checking synchronous sequential circuits that are able to recover after an occurrence of soft errors is proposed.

A fault-tolerant system that is based on two replicas of a self-checking circuit and on error-masking interface has been suggested in [8]. They use two checkers and rather complicate error-masking interface containing flip-flops.

In the paper [9] was suggested a fault-tolerant sequential circuit design also based on two self-checking circuits for soft Path Delay Faults (PDFs) of the circuit. It includes two self-checking circuits, one self-testing checker and more simple error-masking interface than one in [8].

In [10] a fault-tolerant scheme based on totally self-checking system with low overhead in comparison with architectures suggested in [9] is considered. In contrast with these schemes it has only one self-checking combinational circuit and another circuit is conventional one. Such scheme implements the correct behavior of a combinational circuit when any permissible (among SAFs) soft fault (transient or intermittent) occurs. The reliability of the proposed scheme is higher than TMR systems or fault-tolerant systems based on two self-checking circuits.

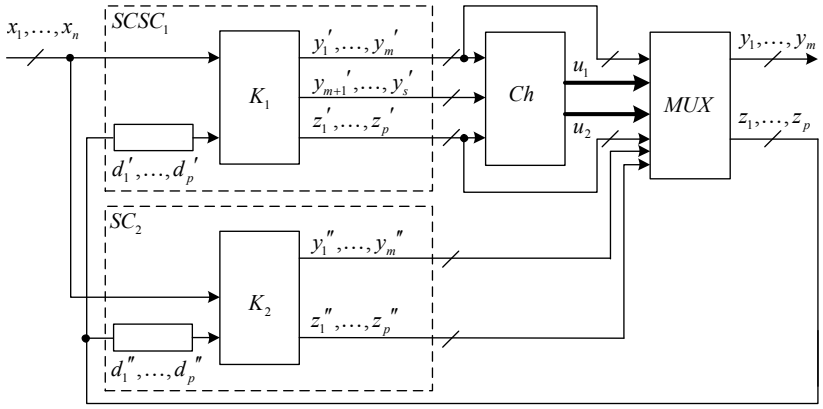

Fig. 1. Fault-tolerant scheme

In this paper we propose a fault-tolerant sequential circuit design based on architecture suggested in [10]. The scheme consists of one self-checking sequential circuit, one normal (unprotected) sequential circuit, not self-testing checker and multiplexor. Such scheme implements the correct behavior of a sequential circuit when any permissible (among SAFs and PDFs) transient or intermittent fault occurs.

This paper is structured as follows. In Section 1 the fault-tolerant scheme construction is described. Section 2 gives the analysis of the fault-tolerant properties for suggested scheme. In section 3 some estimations of the scheme complexity are discussed.

1. Fault-tolerant architecture

We have the State Transition Graph (STG) description of a synchronous finite state machine behavior. It is necessary to derive sequential circuit masking single stuck-at faults at gate poles of the circuit and delay faults.

We assume that each fault is transient or intermittent, and a next fault appears after a previous one has disappeared, and only one module of the fault-tolerant architecture can be faulty.

We suggest applying the architecture of fault-tolerant scheme [10] for sequential circuits. The architecture is based on using only one self-checking circuit.

The implementation of a fault-tolerant sequential circuit shown in Fig. 1.

Here $SCSC_1$ is a self-checking sequential circuit. Assume that all outputs (primary output lines and next state lines) of combinational part (K_1) of sequential circuit are under observation.

We may use any technique providing the unidirectional error manifestation for combinational part of sequential circuit. Single stuck-at faults in combinational part of sequential circuit can be detected if results in unidirectional errors at the outputs and the outputs are encoded using a unidirectional error detecting code, for

example, (k, l)-code (here l – length of code word and k – weight of code word) or Berger code. One of that technique is suggested in [11].

Here we consider the technique described in [12]. We encode FSM states with (k, l)-code words and then change each 0 value in a code word for don't care. As symbols of an output alphabet have already been encoded we add output variables to encode sequential circuit outputs also by the proper (k, l)-code words.

This encoding provides the unidirectional manifestation of single stuck-at faults at gate poles of a combinational part of sequential circuit when using the proper circuit design.

Let we have STG description of FSM represented by Table 1.

Table 1

STG description of FSM

$x_1x_2x_3$ (input cubes)	q (current state)	q' (next state)	$y_1y_2y_3y_4y_5$ (output vectors)
0 – –	1	1	0 0 0 1 0
– 0 –	1	1	0 0 0 1 0
1 1 –	1	2	1 0 0 1 0
– – 0	2	2	0 0 1 1 0
– – 1	2	3	1 0 1 1 0
1 0 –	3	3	0 1 0 0 0
0 – –	3	4	1 1 0 0 0
– 1 –	3	4	1 1 0 0 0
– – 0	4	4	0 1 0 0 1
– – 1	4	1	1 1 0 0 1

After encoding states ((1, 4)-code) and outputs ((3, 7)-code) we get Table 2 representing the system of incompletely specified Boolean functions.

Table 2

System of incompletely specified Boolean functions

$x_1x_2x_3$	$z_1z_2z_3z_4$	$z'_1z'_2z'_3z'_4$	$y_1y_2y_3y_4y_5y_6y_7$
0 – –	1 0 0 0	1 0 0 0	0 0 0 1 0 1 1
– 0 –	1 0 0 0	1 0 0 0	0 0 0 1 0 1 1
1 1 –	1 0 0 0	0 1 0 0	1 0 0 1 0 1 0
– – 0	0 1 0 0	0 1 0 0	0 0 1 1 0 0 1
– – 1	0 1 0 0	0 0 1 0	1 0 1 1 0 0 0
1 0 –	0 0 1 0	0 0 1 0	0 1 0 0 0 1 1
0 – –	0 0 1 0	0 0 0 1	1 1 0 0 0 0 1
– 1 –	0 0 1 0	0 0 0 1	1 1 0 0 0 0 1
– – 0	0 0 0 1	0 0 0 1	0 1 0 0 1 1 0
– – 1	0 0 0 1	1 0 0 0	1 1 0 0 1 0 0

Changing 0 values for don't cares we get minimized system of Boolean functions (Table 3). This changing keeps FSM behavior and originates partially monotonous system of Boolean functions [12].

Table 3

Representation of partially monotonous system of Boolean functions

$x_1x_2x_3$	$z_1z_2z_3z_4$	$z'_1z'_2z'_3z'_4$	$y_1y_2y_3y_4y_5y_6y_7$
0––	1 – – –	1 0 0 0	0 0 0 1 0 1 1
–0–	1 – – –	1 0 0 0	0 0 0 1 0 1 1
11–	1 – – –	0 1 0 0	1 0 0 1 0 1 0
––0	– 1 – –	0 1 0 0	0 0 1 1 0 0 1
––1	– 1 – –	0 0 1 0	1 0 1 1 0 0 0
10–	– – 1 –	0 0 1 0	0 1 0 0 0 1 1
0––	– – 1 –	0 0 0 1	1 1 0 0 0 0 1
–1–	– – 1 –	0 0 0 1	1 1 0 0 0 0 1
––0	– – – 1	0 0 0 1	0 1 0 0 1 1 0
––1	– – – 1	1 0 0 0	1 1 0 0 1 0 0

Derive from Table 3 the system F of partially monotonous (on state variables) functions. For that include cubes of Table 3 marked with 1 value in the same column in the sum of products (SoP) of the corresponding function. The system F for example from Table 3:

$$z'_1 = \overline{x_1} z_1 \vee \overline{x_2} z_1 \vee x_3 z_4,$$

$$z'_2 = x_1 x_2 z_1 \vee \overline{x_3} z_2,$$

$$z'_3 = x_3 z_2 \vee x_1 \overline{x_2} z_3,$$

$$z'_4 = \overline{x_1} z_3 \vee x_2 z_3 \vee \overline{x_3} z_4,$$

$$y_1 = x_1 x_2 z_1 \vee x_3 z_2 \vee \overline{x_1} z_3 \vee x_2 z_3 \vee x_3 z_4,$$

$$y_2 = x_1 \overline{x_2} z_3 \vee \overline{x_1} z_3 \vee x_2 z_3 \vee \overline{x_3} z_4 \vee x_3 z_4,$$

$$y_3 = \overline{x_3} z_2 \vee x_3 z_2,$$

$$y_4 = \overline{x_1} z_1 \vee \overline{x_2} z_1 \vee x_1 x_2 z_1 \vee \overline{x_3} z_2 \vee x_3 z_2,$$

$$y_5 = \overline{x_3} z_4 \vee x_3 z_4,$$

$$y_6 = \overline{x_1} z_1 \vee \overline{x_2} z_1 \vee x_1 x_2 z_1 \vee x_1 \overline{x_2} z_3 \vee \overline{x_3} z_4,$$

$$y_7 = \overline{x_1} z_1 \vee \overline{x_2} z_1 \vee \overline{x_3} z_2 \vee x_1 \overline{x_2} z_3 \vee \overline{x_1} z_3 \vee x_2 z_3.$$

We apply a multilevel synthesis method from [13]:

1. It is based on dividing of SoPs (system F) to get more simple system F^*.

2. Each function (it is presented as SoP in general case) of system F^* is represented by sub-circuit from two inputs NAND gates and NOT gates. All these sub-circuits form virtual circuit named Subject graph.

3. Subject graph may be covered by sub-graphs corresponding to gates of any real gate basis during technology mapping.

It is proved that any single stuck-at fault at gate pole of the Subject graph and also at gate pole of the circuit in any real gate basis obtained in above mentioned way from system F manifests itself on circuit outputs as unidirectional one. This means that the fault changes the values of a sub-set of circuit outputs from 0 to 1 (from 1 to 0) on a test pattern. (If one circuit output changes its value from 1 to 0 then another output cannot change its value from 0 to 1.) Thus we get from table like Table 3 self-checking combinational part of sequential circuit.

Considering sequential circuit as a whole (Fig. 1) we admit single stuck-at faults on flip-flop poles. These faults also manifest themselves as unidirectional ones. Actually, as we use above mentioned synthesis method we get circuit keeping the system F. We mean that if we move from the circuit outputs to inputs substituting instead of internal circuit variables the proper gate functions (depending on gate inputs), so that any simplification of Boolean algebra is forbidden, we get exactly system F. As F is partially monotonous on state variables stuck-at 1 fault at a pole of flip-flop originates increasing on-sets of some functions from F, single stuck-at 0 fault at pole of flip-flop originates decreasing on-sets of some functions from F. Thus the considered faults manifest themselves as unidirectional one. It means that $SCSC_1$ is self-checking sequential circuits for single stuck-at faults at gate poles of its combinational part and the same faults at flip-flop poles.

SC_2 is a sequential circuit realizing STG description of FSM. It has the same encoding states (like $SCSC_1$), but has no additional outputs that are used for providing unidirectional error detection. In the example considered (Table 3) the additional outputs are marked with y_6, y_7. K_2 is a combinational part of sequential circuit SC_2, implementing the system of partially monotonous Boolean functions (without functions corresponding to additional outputs). The circuit is derived by using any method that provides low cost realization. For increasing reliability properties of the system it is desirable applying different synthesis methods for $SCSC_1$ and SC_2. Such approach allows decreasing a probability of appearance of identical faults [14].

Variables $y'_1,...,y'_m$, ($y''_1,...,y''_m$) correspond to outputs of sequential circuits; variables $y'_{m+1},...,y'_{m+s}$ correspond to additional outputs for $SCSC_1$ providing (k, l)-code words; variables $z'_1,...,z'_p$ are state ones, and $d'_1,...,d'_p$ ($d''_1,...,d''_p$) are flip-flops corresponding to these variables.

Ch is an arbitrary (k, l)-code checker. It may be not self-testing. It is supposed that each next fault appears when a previous one has disappeared. If checker fault manifests itself, it is masked by correct outputs of SC_2. This fault does not effect on any next transient or intermittent fault.

The checker detects erroneous code words on outputs of K_1: $y'_1,...,y'_m$, $y'_{m+1},...,y'_{m+s}$, $z'_1,...,z'_p$.

Here we suggest using not self-testing monotonous checker that has low cost realization. The checker has two outputs (u_1, u_2): "10" if both $SCSC_1$ and Ch are error-free; "00", "01" and "11" if an error in one of these circuits is detected.

MUX is a multiplexor with control inputs u_1, u_2 and data inputs $y'_1,...,y'_m$, $z'_1,...,z'_p$, $y''_1,...,y''_m$, $z''_1,...,z''_p$. The MUX connects lines $y'_1,...,y'_m$, $z'_1,...,z'_p$ with lines $y_1,...,y_m$, $z_1,...,z_p$ when checker outputs have "10" values otherwise the MUX connects lines $y''_1,...,y''_m$, $z''_1,...,z''_p$ with lines $y_1,...,y_m$, $z_1,...,z_p$.

2. Fault-tolerance analysis

We consider single stuck-at faults at gate poles of the combinational parts of sequential circuit $SCSC_1$ and its flip-flops, and single path delay faults of $SCSC_1$. As for SC_2 and Ch their faults may be arbitrary but any fault keeps circuit as combinational one. All above mentioned faults must be transient or intermittent, and a fault occurs one at a time and a next fault from permissible set can appear only after a forgoing fault has disappeared. Only one circuit among $SCSC_1$, SC_2, Ch, MUX may be faulty.

2.1. Stuck-at faults

Notice as V_{SCSC_1} a set of permissible faults of $SCSC_1$. V_{SCSC_1} consists of single stuck-at faults at gate poles and single stuck-at faults at inputs and outputs of flip-flops. All these faults manifest themselves as unidirectional ones on outputs of sub-circuit K_1. In the presence of any fault from V_{SCSC_1} circuit $SCSC_1$ may produce non-code word at the outputs of combinational part K_1 that will be detected by checker and multiplexer will use erroneous free outputs from K_2.

Let V_{CV} be a set of arbitrary faults of the checker. The fault-free checker for code words on inputs generates signals "10", for non-code words – "00", "01", "11". In presence of any fault from V_{CV} the checker can produce arbitrary signals ("00", "11", "01", "10") at the outputs that drives multiplexer switching between error free outputs from K_1 or K_2.

Let V_{SC_2} be a set of arbitrary faults of circuit SC_2. It is supposed only one module of the system may be faulty. This means that other modules are fault-free and the multiplexor uses error free outputs from K_1.

Let V_{MUX} be a set of permissible faults of the multiplexer. These faults can change connection of some lines $y'_1,...,y'_m$, $z'_1,...,z'_p$ for corresponding lines $y''_1,...,y''_m$, $z''_1,...,z''_p$. In this case K_1 and K_2 are fault free and both have error free outputs.

Faults on primary inputs $(x_1, x_2,..., x_n)$ and primary outputs $(y_1, y_2,..., y_m)$ are not considered.

Note $V = V_{SCSC_1} \cup V_{Ch} \cup V_{SCSC_2} \cup V_{MUX}$.

Proposal 1. The scheme of Fig. 1 keeps correct functioning in the presence of any fault from *V*.

2.2. Path delay faults

Consider a combinational circuit in which at time moment *t* vector v_1 of values of input variables of a circuit is replaced by another vector v_2. Let τ be a maximal admissible path delay in the circuit. If in time peri-

od τ after the time moment *t*, the expected value of vector v_2 on the circuit output does not appear, we say that the circuit has path delay faults for some paths. We say that a pair (v_1, v_2) detects such fault, and the fault manifests itself on this pair.

Take into consideration, that path delays for opposite (inverse) changes of signal values on outputs of path gates may be different.

We call a pair that detects a delay of a signal on a circuit output changing from 0 to 1 as a test for a rising transition; a delay of a signal on the circuit output changing from 1 to 0 as a test for a falling transition.

They distinguish single and multiple path delay faults, meaning faults of one or several paths, but we consider only single PDFs.

If a test pair detects the path delay fault regardless of delay faults of other paths of a circuit, we call this pair a robust test for this fault and the fault itself is called robust testable.

If a test pair detects the path delay fault in the considered path only on the assumption that other paths of a circuit are fault-free, this pair is called a non-robust test for the fault, and the fault itself is called non robust testable.

It's possible to reduce construction of a test pair for a PDF to testing the constant fault in the corresponding literal of the equivalent normal form (ENF) originated by the sub-circuit that contains the path considered. Dealing with a single literal fault we mean either turning the literal to constant 1 which leads to this literal disappearing from ENF products (*bp*-fault) or turning the literal to constant 0 which leads to disappearing all products that contain the literal (*ap*-fault).

A test pattern for this literal is vector v_2 of a test pair. Note that a PDF manifests itself on this test pattern only if previous vector v_1 differs from v_2 by a value of the variable marking the beginning of this path and possibly values of other variables. Otherwise this PDF does not manifest itself on the circuit output.

Note that only vector v_1 in a test pair indicates if the test pair is robust or non-robust. This vector does not directly affect PDF manifestation; it only provides manifestation of the PDF on vector v_2.

Thus a PDF (robust and non-robust testable) differs from the corresponding literal constant fault by manifestation not on each test pattern for the literal.

Take into consideration that disappearing of literal from ENF products increases on-set of the sub-circuit function, and disappearing of ENF products decreases on-set of the sub-circuit function. This means that a literal constant fault manifests itself in a combinational part K_1 of the scheme as a single stuck-at fault of this sub-circuit, but on the only sub-circuit output.

Masking PDFs we don't need to know what path is fault. That is why we do not differ robust testable and non-robust testable PDFs and, consequently, we have to pay attention only to the literal originated by the path.

Note that for any path opposite delays of signals are feasible at the same time. Test patterns for constant 1 and constant 0 fault of the proper ENF literal are different. Consequently, opposite delays of the same path manifest themselves on different input vectors.

Thus when PDF of a falling transition occurs, we observe 1 value instead of 0 value on the certain test pattern for *bp*-fault, and for a rising transition – 0 value instead of 1 value on the certain pattern for *ap*-fault.

As we consider transient or intermittent PDFs we may observe these faults only during time *T*, $T \geq \tau$. During time *T* path delay may manifest itself several times both for rising and falling transitions, but only on the same circuit output.

Let $V_{PDF} = V_{SCSC_1}^{PDF} \cup V_{Others}^{PDF}$. Here $V_{SCSC_1}^{PDF}$ is a set of single path delay faults in $SCSC_1$. These faults manifest themselves changing only one output of the combinational part K_1, and a non-code word appears on inputs of the checker. V_{Others}^{PDF} is a set of path delay faults in checker (Ch) and K_2. If Ch has a PDF then combinational parts K_1 and K_2 are fault free. If K_2 has a PDF then K_1 is fault free. Both cases provide correct outputs.

Proposal 2. The scheme of Fig. 1 keeps correct functioning in the presence of any fault from V_{PDF}.

3. Experimental results

For experimental results we use MCNC sequential benchmarks [15, 16]. Appreciating a complexity of the schemes we do not consider complexities of MUXs, voting circuits and flip-flops.

For state assignment of the TMR scheme we use the encoding with the shortest code words. For encoding states of the FSMs ($SCSC_1$ and SC_2) is used one-hot encoding.

Encoding outputs of a self-checking sequential circuit we add minimal number of output variables to get the proper (*k*, *l*)-code.

For deriving structural realization of the circuits (K_1 and K_2) we use a multilevel synthesis method [13] based on dividing of SoPs and using two inputs NAND gates and NOT gates. For synthesis of the not self-checking monotonous two inputs AND and OR gates are used. The complexity of schemes and their sub-circuits are estimated by the number of NOT gates and two inputs gates.

Table 4 illustrates the area overhead reduction of the proposed scheme as compared to TMR scheme and scheme from [9]. Use the following notations:

Table 4

An estimation of the complexity of different fault-tolerant schemes

Name	*n*	*m*	*s*	L_{TMR}	L_{2SC}	L_{1SC}	HR_{TMR} (%)	HR_{2SC} (%)
bbara	4	2	10	714	449	418	41,46	6,9
bbtas	2	2	6	300	325	242	19,33	25,54
beecount	3	4	7	471	427	405	14,01	5,15
dk14	3	5	7	1149	992	891	22,45	10,18
dk16	2	3	27	2202	1510	1198	45,59	20,66
dk17	2	3	8	567	436	368	35,1	15,6
dk27	1	2	7	216	142	110	49,07	22,54
dk512	1	3	15	558	323	255	54,3	21,05
donfile	2	1	24	1965	917	867	55,88	5,45
ex2	2	2	19	1209	794	618	48,88	22,17
ex3	2	2	10	483	474	350	27,54	26,16
ex5	2	2	9	414	370	298	28,02	19,46
ex6	5	8	8	1032	786	644	37,6	18,07
ex7	2	2	10	528	420	332	37,12	20,95
s1	8	6	20	2754	2213	1876	31,88	15,23
s1494	8	19	48	8121	7311	8635	9,98	15,33
s208	11	2	18	2091	1696	1451	30,61	14,45
s420	19	2	18	2124	1758	1452	31,64	17,41
s510	19	7	47	2244	2275	1857	17,25	18,37
s820	18	19	25	4371	3956	3397	22,28	14,13
s832	18	19	25	4929	4458	3794	23,03	14,89
shiftreg	1	1	8	288	150	118	59,03	21,33
tma	7	6	20	1290	1293	1072	16,9	17,09
train11	2	1	11	501	281	241	51,9	14,23
Average							**33,79**	**16,76**

Note. Name – the name of benchmark circuit; *n* – the number of inputs; *m* – the number of outputs; *s* – the number of states; L_{TMR} – the number of gates for TMR scheme; L_{2SC} – the number of gates for fault-tolerant scheme based on two self-checking sequential circuits, L_{1SC} – the number of gates for the proposed fault-tolerant scheme; HR_{TMR} – the area overhead reduction of the proposed scheme as compared to TMR; HR_{2SC} – the area overhead reduction of the proposed scheme as compared to scheme from [9].

As seen from the last two columns, the proposed method provides an average saving of 33,79% concerning TMR implementation and 16,76% concerning method described in [Ibid.].

Conclusion

In this paper we have proposed a fault-tolerant sequential circuit design based on a self-checking system with low overhead. The suggested scheme masks not only transient (intermittent) single stuck-at faults at gate

poles but path delay faults as well. The experimental results show that the proposed technique provides an average saving of 33,79% overhead concerning TMR implementation and 16,76% overhead concerning previous technique based on two self-checking modules.

REFERENCES

1. Von Neumann, J. (1958) Probabilistic logics and the synthesis of reliable from unreliable components. In: Shannon, C. (ed.) *Automata Studies*. Princeton, NJ: Princeton University Press. pp. 43–98.
2. Pradhan, D.K. (1996) *Fault-tolerant computer system design*. Upper Saddle River, N.J: Prentice Hall PTR.
3. Abramovici, M., Breuer, M.A. & Friedman, A.D. (1994) *Digital systems testing and testable design*. New York: IEEE Press.
4. Lyons, R.E. & Vanderkulk, W. (1962) The Use of Triple-Modular Redundancy to Improve Computer Reliability. *IBM Journal of Research and Development*. 2(6). pp. 200–209. DOI: 10.1147/rd.62.0200
5. El-Maleh, A.H. & Al-Qahtani, A.S. (2014) A finite state machine based fault tolerance technique for sequential circuits. *Microelectron. Rel.* 54(3). pp. 654–661. DOI: 10.1049/iet-cdt.2016.0085
6. Cassel, M. & Kastensmidt, F.L. (2006) Evaluating one-hot encoding finite state machines for SEU reliability in SRAM-based FPGAs. IOLTS'2006. Lake of Como, Italy, July. *Proc. of the 12th IEEE Int. On-line Testing Symp.* pp. 139–44.
7. Levin, I., Matrosova, A. & Ostanin, S. (2001) Survivable Self-checking Sequential Circuits. DFT'2001. *Proc. of the IEEE Int. Symp. Defect and Fault Tolerance in VLSI Systems*. San Francisco, USA, October 2001. pp. 395–402.
8. Lubaszewski, M. & Courtois, B. (1998) A reliable fail-safe system. *IEEE Tran. On Comp.* 47(2). pp. 236–241.
9. Matrosova, A., Ostanin, S., Kirienko, I. & Nikolaeva, E. (2015) Fault-tolerant High Performance Scheme Design. EWDTS'2015. *Proc. of IEEE East-West Design & Test Symp.* Batumi, Georgia, Sept. 2015. pp. 286–289.
10. Ostanin, S., Kirienko, I. & Lavrov, V. (2015) Fault-Tolerant Combinational Circuit Design. EWDTS'2015. *Proc. of IEEE East-West Design & Test Symp.* Batumi, Georgia, Sept. pp. 302–305.
11. Busaba, F. & Lala, P. (1994) Self-Checking Combinational Circuit Design for Single and Uniderectional Multibit Error. *JETTA*. 5. pp. 19–28.
12. Matrosova, A. Yu. & Ostanin, S.A. (1998) Self-Checking Synchronous FSM network Design. *Proc. of the 4th IEEE Int. On-line Testing Workshop*. IOLTS'1998. Capri, Italy, July 1998. pp. 162–166.
13. Murgai, R., Brayton, R. & Sangiovanni-Vincentelli, A. (1995) *Logic Synthesis for Field Programmable Gate Arrays*. Boston, Mass.: Kluwer Academic Publishers.
14. Mitra, S., Saxena, N.R. & McCluskey, E.J. (2002) A Design Diversity Metric and Analysis of Redundant Systems. *IEEE Trans. Comput.* 51(5). pp. 498–510.
15. Yang, S. (1991) *Logic Synthesis and Optimization Benchmarks User Guide Version 3.0*.
16. Berkeley Logic Synthesis and Verification Group. (2017) *ABC: A System for Sequential Synthesis and Verification*. [Online] Available from: http://www.eecs.berkeley.edu/~alanmi/abc/.

***Matrjsova Anjela Yurievna*,** Dr. Sc., Professor. E-mail: mau11@yandex.ru
***Ostanin Sergey Alexandrovich*,** Cand. Sc., Associate Professor. E-mail: sergeiostanin@yandex.ru
***Kirienko Irina Evgenievna*.** E-mail: irina.kirienko@sibmail.com
***Nikolaeva Ekaterina Alexandrovna*,** Cand. Sc. E-mail: nikolaeve-ea@yandex.ru
National Research Tomsk State University

Поступила в редакцию 12 июня 2017 г.

Матросова Анжела Юрьевна, Останин Сергей Александрович, Кириенко Ирина Евгеньевна, Николаева Екатерина Александровна (Томский государственный университет, Российская Федерация).

Проектирование отказоустойчивых последовательностных схем для константных неисправностей и неисправностей задержек путей.

Ключевые слова: отказоустойчивая схема; последовательностная схема; самопроверяемая схема; константная неисправность; неисправность задержки пути; нерегулярная ошибка.

Предлагается метод синтеза отказоустойчивых последовательностных схем, основанный на использовании самопроверяемой схемы и схемы без дополнительных свойств, реализующей основную функциональность, для одиночных константных неисправностей и неисправностей задержек путей, кратковременных или перемежающихся. Отказоустойчивые свойства предложенной архитектуры обеспечиваются в предположении, что в текущий момент времени может быть неисправен только один функциональный блок. Рассматриваются кратковременные и перемежающиеся неисправности, причем допускается появление очередной неисправности только после исчезновения последствий от предыдущей неисправности. Доказывается, что проявление одиночных константных неисправностей и неисправностей задержек путей в предложенной архитектуре маскируется в рамках заданных ограничений, т.е. схема является отказоустойчивой. Экспериментальные результаты показали, что предложенный метод позволяет сократить аппаратурную избыточность в среднем на 33,79% по сравнению с методом тройного резервирования модулей и на 16,76% по сравнению с отказоустойчивыми схемами, основанными на использовании двух самопроверяемых модулей.