Survivable Self-checking Sequential Circuits[1]

I. Levin
Tel-Aviv University, Israel
ilia1@post.tau.ac.il

A. Matrosova
Tomsk State University, Russia
mau@fpmk.tsu.ru

S. Ostanin
Tel-Aviv University, Israel
sostanin@post.tau.ac.il

Abstract

This paper presents a method for designing totally self-checking synchronous sequential circuits (SSC), and investigates their behavior in presence of transient faults. We deal with the case when the circuit is able to recover after the number of clocks. We call SSC owing this property as a survivable SSC.

A concept of a partially monotonous SSC is developed in the paper. It is proven that the partially monotonous SSCs are survivable.

1. Introduction

Known accepted approaches to synthesis of self-checking synchronous sequential circuits (SSC) are based on the following definitions.

Definition 1. A sequential circuit is self-testing if, for every fault in a fault set, there is an input/state code pair in the circuit such that a non-code output is produced.

Definition 2. A sequential circuit is fault secure if, for every fault from the faulty set the sequential circuit never produces an incorrect code output for a code input.

A sequential circuit is totally self-checking (TSC) if it is both self-testing and fault-secure.

The problem of designing totally self-checking sequential circuits has been examined in [1-3]. The paper [1] describes a procedure for designing the Moore-type sequential circuits. States of the circuit are coded by using a *k-out-of-n* code, and the on-set realization of outputs is used. The self-testing properties are obtained by removing untested inputs. Adding a self-checking checker to the outputs produces a TSC realization.

A method for designing totally self-checking Mealy-type state machines (sequential circuits) is presented in [2]. It uses *1-hot* code for states and inputs assignment. Outputs are coded by any unordered code.

The paper [2] states that for detecting a fault by test vector, the following two Conditions are to be satisfied:

1. $\forall(S_k, I_j)$, $\exists(S_i, I_j)$, $i \neq k$, such that $Z(S_i, I_j) \neq Z(S_k, I_j)$;
2. $\forall(S_k, I_j)$, $\exists(S_k, I_i)$, $i \neq j$ such that $Z(S_k, I_i) \neq Z(S_k, I_j)$

[1]This research was supported by BSF under grant No. 9800154.

(S_k, S_i - states of a sequential circuit, I_j, I_i - input vectors, Z – output functions).

If the self-checking SSC includes a separate memory checker, any fault will be detected either immediately by the output checker, or on the next clock by the memory checker [3].

If the memory checker is absent in the SSC checking scheme, the straightforward implementation based on redefining of all SSC functions by "zeros" on all non-code state words can be applied. In this case a null-vector will be produced on the SSC's outputs and obviously will be detected immediately as a non-code output vector.

On the one hand, above solutions are very different. The distinction between the solutions can be considered as a trade-off between the SSC hardware overhead and the complexity of the memory checker. On the other hand, the solutions have the equal functionally since both of them require of immediate (or next clock) detection of a fault.

This requirement is highly reasonable for permanent faults since such faults have to be detected as soon as possible. However, in the case of transient faults the approach may be different. According to Definition 2 the sequential circuits may pass through several incorrect states, while maintaining correct outputs before an error indication. Such a situation suits to the transient faults since before the error indication, both the fault and its consequences may disappear, and the SSC may become fault-free again. Would we use the strict approach of [2, 4-7], we would prematurely mark as erroneous a SSC, which could successfully survive.

To the best of our knowledge the known works did not try to design TSC SSCs in which above-mentioned requirement of the "urgent" fault detection does not satisfied. This requirement do not allow "saving" the SSC capable of recovering from transient faults and their consequences. Exactly this case will be in the focus of the present paper. The SSC that is able to survive after occurrence of a transient fault will be called a *survivable SSC*.

To formulate our approach, let us first define four different modes of functioning of the SSC and describe its behavior in presence of either permanent or transient faults, using a graphical representation. We introduce a *Mode Transition Graph* (MTG) for this purpose. Vertexes of the MTG correspond to specific modes of the circuit. Transitions between the modes correspond to specific events.

MTG describing the proposed concept with respect to a permanent fault is shown in Figure 1.

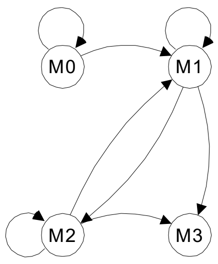

Figure 1. Mode Transition Graph for a permanent fault

M_0 - Fault free mode. The circuit remains in the fault free mode until a fault occurs.

M_1 - Latent mode. It is a mode where the presence of a fault cannot be detected concurrently with the SSC functioning since the test vector detecting the fault has not yet appeared on the circuit's inputs. The circuit moves to the latent mode from the fault free mode when a fault occurs, and leaves the latent mode when the test vector is applied to the circuit.

M_2 - Silent mode. It is a mode where the presence of a fault does not manifest itself in a form of non-code output, while the presence of the fault could potentially be detected by adding of specific observability points. The circuit moves to the silent mode from the latent mode when a non-code state vector appears on the flip-flop inputs or outputs. From the fault secure mode the circuit is able either to move to an erroneous mode (**M_3**), or to return back to the latent mode.

M_3 - Erroneous mode. It is a mode where the circuit terminates its proper functioning, i.e. where a non-code output vector has been produced. The circuit is able to move to this mode either from the silent mode, or from the latent mode.

The Mode Transition Graph for a transient fault is shown in Figure 2.

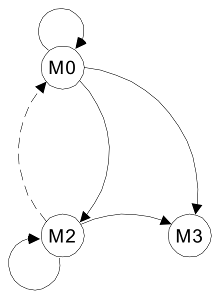

Figure 2. Mode Transition Graph for a transient fault

Obviously, the latent mode **M_1** is absent in the case of a transient fault. When the transient fault occurs, the SSC moves from **M_0** either to the Erroneous mode **M_3**, (if a non-code output vector is produced) or to the silent mode **M_2**, if a non-code next state vector is produced, while the output vector is a codeword. Note, that the sequential circuit is able to return back to the **M_0** mode after its functioning in the **M_2**, which means that the SSC has become fault free again, that is survives.

The rest of the paper is organized as follows. Section 2 introduces the partially monotonous system and some other definitions. Section 3 defines the concept of A-, B-faults. Influence of transient faults to behavior of the SSC and all necessary proofs are presented in Section 4. Conclusions are presented in Section 5.

2. Definitions and Assumptions

We propose to implement a self-checking SSC design using the basic scheme shown in Figure 3.

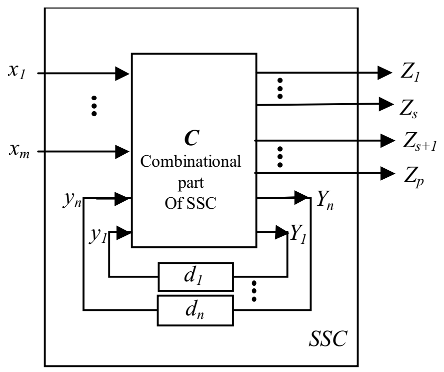

Figure 3. Basic scheme of the SSC

According to this scheme, the synchronous sequential circuit to be checked consists of three portions: an output functions portion (its outputs are $\{Z_1,...,Z_s\}$), a redundant portion (its outputs are $\{Z_{s+1},...,Z_p\}$) that provides encoding of the SSC outputs, and a transition functions portion ($Y=\{Y_1,...,Y_n\}$) that represents the next states. The output functions portion and the redundant portion (its outputs are $Z=\{Z_1,...,Z_s,Z_{s+1},...,Z_p\}$) together form codewords belonging to the Berger code or to the constant weight code.

In Figure 3, C is a combinational part of the SSC, $d_1,...,d_n$ are d-flip-flops.

The combinational circuit C implements the system F of $p+n$ Boolean functions $\mathrm{O}=\{Y_1,...,Y_n\}\cup\{Z_1,...,Z_p\}$ on $m+n$ Boolean variables $\mathrm{I}=\{x_1,...,x_m\}\cup\{y_1,...,y_n\}$.

Let α,β are minterms of the system F represented by Boolean vectors of the length $m+n$. α_i,β_i are components of α,β correspondingly. If $\alpha_i \le \beta_i$ for $i=\overline{(1,(m+n))}$, then β covers α ($\alpha \le \beta$).

Let $\tilde{\alpha},\tilde{\beta}$ are different sequences of Boolean vectors. The sequences $\tilde{\alpha},\tilde{\beta}$ are comparable if for each pair α^i,β^i (α^i,β^i are i-th components of $\tilde{\alpha},\tilde{\beta}$ correspondingly) of their members we have: either $\alpha^i \le \beta^i$ or $\alpha^i \ge \beta^i$. Notice it as $\tilde{\alpha} \lessgtr \tilde{\beta}$.

The system F is monotonous if for any pair of Boolean vectors α,β of the length $m+n$ so that $\alpha \le \beta$, the condition $F(\alpha)\le F(\beta)$ is satisfied. For example, the following system is a monotonous one:

$$\begin{cases} f_1 = x_1 \vee x_3 \vee x_5 \vee x_6 \\ f_2 = x_1 x_2 \end{cases}.$$

Consider a subset I^* of Boolean variables, $\mathrm{I}^* \subseteq \mathrm{I}$. Let α,β be Boolean vectors of $m+n$ variables. If for any component i corresponding to the variable from I^*, $\alpha_i \le \beta_i$, while for any other component j corresponding to the variable $\{\mathrm{I}\setminus\mathrm{I}^*\}$, $\alpha_j=\beta_j$, then β covers α in I^* variables ($\alpha \overset{\mathrm{I}*}{\le} \beta$). For example, $\mathrm{I}^*=\{x_1,x_2,x_3,x_4\}$, $\alpha_1=100001$, $\alpha_2=101101$, $\alpha \overset{\mathrm{I}*}{\le} \beta$.

The system F is partially monotonous in I^* variables if for any pair of Boolean vectors α,β so that $\alpha \overset{\mathrm{I}*}{\le} \beta$, the condition $F(\alpha)\le F(\beta)$ is satisfied.

For example, the system

$$\begin{cases} f_1 = x_1 \vee x_3 \vee x_4 \vee \bar{x}_5 \vee \bar{x}_6 \\ f_2 = x_1 x_2 \end{cases}$$

is partially monotonous in the subset $\mathrm{I}^* = \{x_1, x_2, x_3, x_4\}$.

If for any pair α^i, β^i of members of sequences $\tilde{\alpha}, \tilde{\beta}$ the condition $\alpha^i \overset{\mathrm{I}*}{\leq} \beta^i$ is satisfied, then sequence $\tilde{\beta}$ covers $\tilde{\alpha}$ in I^* variables ($\tilde{\alpha} \overset{\mathrm{I}*}{\leq} \tilde{\beta}$).

Notice that if F is monotonous, it is partially monotonous in any subset of its variables.

Consider two different systems F_1 and F_2 of the same number of functions and variables. If for any α $F_1(\alpha) \leq F_2(\alpha)$ then $F_1 \prec F_2$, F_2 implicates F_1.

Consider two different systems F_1 and F_2 to be partially monotonous in I^* variables. They both depend on the same number of variables and consist of the same number of functions, moreover $F_1 \prec F_2$ (F_2 implicates F_1). For example, we have systems F_1 and F_2:

$$F_1 = \begin{cases} f_1 = x_1 \vee x_3 \vee x_4 \vee \bar{x}_5 \vee \bar{x}_6 \\ f_2 = x_1 x_2 \end{cases}$$

$$F_2 = \begin{cases} f_1 = x_1 \vee x_3 \vee x_4 \vee \bar{x}_5 \vee \bar{x}_6 \\ f_2 = x_1 \end{cases}$$

They are both partially monotonous in $\mathrm{I}^* = \{x_1, x_2, x_3, x_4\}$ variables and F_2 implicates F_1 ($F_1 \prec F_2$).

If F_1 and F_2 are partially monotonous systems and $F_1 \prec F_2$ we have the following:

1. If $\alpha \overset{\mathrm{I}*}{\leq} \beta$ then $F_1(\alpha) \leq F_2(\beta)$,
2. If $\tilde{\alpha} \overset{\mathrm{I}*}{\leq} \tilde{\beta}$ then $F_1(\tilde{\alpha}) \leq F_2(\tilde{\beta})$.

Now we present an example of the partially monotonous system. It is an example of the SSC, having the state and output equations being *unate* in state variables and *binate* in primary input variables. This example is shown in Table1.

Table 1. A partially monotonous system for the exemplary SSC

N0	$x_1x_2x_3$	$y_1y_2y_3y_4$	$Y_1Y_2Y_3Y_4$	$Z_1Z_2Z_3Z_4Z_5Z_6Z_7$
1	0 - -	1 - - -	1 0 0 0	0 0 0 1 0 1 1
2	- 0 -	1- - -	1 0 0 0	0 0 0 1 0 1 1
3	1 1 -	1- - -	0 1 0 0	1 0 0 1 0 1 0
4	- - 0	- 1 - -	0 1 0 0	0 0 1 1 0 0 1
5	- - 1	- 1 - -	0 0 1 0	1 0 0 1 0 1 0
6	1 0 -	- - 1 -	0 0 1 0	1 0 0 1 0 1 0
7	0 - -	- - 1 -	0 0 0 1	1 1 0 0 0 0 1
8	- 1 -	- - 1 -	0 0 0 1	1 1 0 0 0 0 1
9	- - 0	- - - 1	0 0 0 1	0 1 0 0 1 1 0
10	- - 1	- - - 1	1 0 0 0	1 1 0 0 1 0 0

3. Faults

Consider a system F which is partially monotonous in the state variables and a certain fault v from a set of faults V. We study the system F, corresponding to the combinational circuit C (the combinational part of SSC). Let F^v is a system corresponding to this fault and F^v is also partially monotonous in the state variables. If $F^v \prec F$ (F implicates F^v), let us call the fault as A-fault. If $F \prec F^v$ (F^v implicates F), let us call the fault as B-fault.

Include into the set of faults V only A- and B- faults and consider their properties. Obviously, if any fault v is unidirectional, then it is either A-fault or B-fault.

In practice it is useful to distinguish between the A- and B- faults. Considering different synthesis methods and different sets of faults resulted from these methods, we can clear up whether the corresponding set contains only A-, B-faults or not. If the set contains only A-, B-faults we can use the basic scheme shown in Figure 3.

It has been proven in [9] that if the SSC has been synthesized as the partially monotonous system by using either two-level or multi-level methods, then any single stuck-at fault on the gate poles and state lines of the circuit are A-, B- faults.

4. Sequential Circuit Behavior with Respect to Transient Faults

Describe influence of a fault v upon SSC behavior under observing only SSC output lines. Divide the system F for two subsystems φ and ψ. Here φ is the subsystem corresponding to the SSC output functions and ψ is the subsystem corresponding to the SSC next state functions.

Let SSC has an initial state and any input sequence $\tilde{\gamma}$ begins with the reset input. Let $\tilde{\alpha}$ is the input sequence of the combinational circuit C (Figure 3) resulted from $\tilde{\gamma}$ and $\tilde{\alpha}^v$ - the input sequence resulted from $\tilde{\gamma}$ and v. The sequences $\tilde{\alpha}$, $\tilde{\alpha}^v$ can differ from each other only on values of the state variables $(y_1,...,y_n)$ of SSC.

A fault v is detectable on the input sequence $\tilde{\gamma}$ if the fault manifests itself on the SSC output lines as a unidirectional one during the sequence and moreover: $\varphi(\tilde{\alpha}) \geq \varphi^v(\tilde{\alpha}^v)$ or $\varphi(\tilde{\alpha}) \leq \varphi^v(\tilde{\alpha}^v)$, otherwise the fault is undetectable: $\varphi(\tilde{\alpha}) = \varphi^v(\tilde{\alpha}^v)$.

Theorem 1. If A (B) - fault manifests itself on the state lines beginning from the l-th member of the sequence $\tilde{\gamma}$ then $\alpha^{v,l+1} \leq \alpha^{l+1}$ ($\alpha^{l+1} \leq \alpha^{v,l+1}$), $\alpha^{v,l+2} \leq \alpha^{l+2}$ ($\alpha^{l+2} \leq \alpha^{v,l+2}$), … till the end of $\tilde{\gamma}$ and the corresponding $\tilde{\alpha}$, $\tilde{\alpha}^v$.

Proof. As l is the number of a member of the input sequence $\tilde{\gamma}$ on which the fault first manifests itself on the combinational circuit's C output lines, then $F^v(\alpha^l) < F(\alpha^l)$ ($F(\alpha^l) < F^v(\alpha^l)$) since partially monotony property of F and F^v and because of v is A(B) - fault. Consequently, $\psi^v(\alpha^l) < \psi(\alpha^l)$ ($\psi(\alpha^l) < \psi^v(\alpha^l)$) that is $\alpha^{v,l+1} \leq \alpha^{l+1}$ ($\alpha^{l+1} \leq \alpha^{v,l+1}$). Beginning from $l+1$ member and till the μ-th member of the sequence $\tilde{\gamma}$ we have $\psi^v(\alpha^{v,l+\tau}) \leq \psi(\alpha^{v+\tau})$ ($\psi(\alpha^{l+\tau}) \leq \psi^v(\alpha^{v,l+\tau})$), $\tau = 1,...,\mu$, resulted from acting the fault v and the 1, 2 properties of the section 2.

Here $\mu+1$ is a clock on which the fault v terminated its action. Then we have:

$\psi(\alpha^{v,l+\tau}) \leq \psi(\alpha^{l+\tau})$ ($\psi(\alpha^{l+\tau}) \leq \psi(\alpha^{v,l+\tau})$), $\tau = \mu+1, \mu+2,...$ till the end of $\tilde{\gamma}, \tilde{\alpha}, \tilde{\alpha}^v$. It means $\alpha^{v,l+2} \leq \alpha^{l+2}$ ($\alpha^{l+2} \leq \alpha^{v,l+2}$), … till the end of γ and the corresponding α, α^v. Q.E.D.

Theorem 2. If a B-fault manifests itself as a unidirectional one on the combinational circuit C outputs beginning from l member of the sequence $\tilde{\gamma}$, then the fault is either unidirectional or undetectable.

Proof. The proof is based on the $\tilde{\alpha}$, $\tilde{\alpha}^v$ properties established in the Theorem 1. It is possible that during action of B-fault and for the certain τ we will first have: $\varphi(\alpha^{l+\tau}) < \varphi^v(\alpha^{v,l+\tau})$ ($\varphi(\alpha^{l+1}) = \varphi^v(\alpha^{v,l+1}),...,\varphi(\alpha^{l+\tau-1}) = \varphi^v(\alpha^{v,l+\tau-1})$). We restrict $\tilde{\gamma}, \tilde{\alpha}, \tilde{\alpha}^v$

with $(l+\tau)$ members. In this case, the fault is unidirectional. Since the fault action terminates by the certain τ ,than $\varphi(\alpha^{l+\tau}) < \varphi(\alpha^{v,l+\tau})$. The values of the fault free SSC and the faulty SSC output lines are equal for $l,...,l+\tau-1$ members. Otherwise the B-fault is undetectable on the SSC output lines during the sequence $\tilde{\gamma}$. Q.E.D.

Theorem 3. Let B-fault manifests itself during the sequence $\tilde{\gamma}$ on the SSC state lines and remains undetectable on the SSC output lines, and this fault stopped its action during the sequence $\tilde{\gamma}$. Then this fault is undetectable for any input sequence that follows the sequence $\tilde{\gamma}$.

Proof. Since any sequence begins from the reset input and SSC becomes fault free before appearance of a sequence that follows the sequence $\tilde{\gamma}$, then the fault does not manifest itself on such sequence and, consequently, the fault is undetectable on this sequence. Q.E.D.

If B-fault action during the sequence $\tilde{\gamma}$ remains undetectable on the SSC output lines while being manifested on the SSC state lines then the fault has been masked. We will call it as fault masking property. Unfortunately, this property exists only for B-faults.

Theorem 4. If A-fault first manifests itself at the combinational circuit *C* outputs on the *l*-th member of the input sequence $\tilde{\gamma}$, then the fault is unidirectional either on the *l*-th or the $(l+1)$-th member of this sequence.

Proof. We have $F^v(\alpha^l) < F(\alpha^l)$. If $\varphi^v(\alpha^l) < \varphi(\alpha^l)$ the fault manifests itself on the SSC output lines. Otherwise $\psi^v(\alpha^l) < \psi(\alpha^l)$. $\psi(\alpha^l)$ represents a Boolean vector (state codeword) that is orthogonal to any transition of the SSC transition table (representing the system *F*) and that corresponds to another state codeword. It follows from bi-directionality of any pair of state codewords. Replacing 0 values with "don't cares" in state codewords preserves orthogonality of any state codeword to cubes corresponding to other state codewords resulted from such a replacement. $\psi^v(\alpha^l)$ contains more 0-values in comparison with $\psi(\alpha^l)$. Then this Boolean vector is orthogonal to any transition of the SSC transition table. $F^v < F$, consequently, $\psi^v(\alpha^l)$ is orthogonal to any transition of the SSC transition table of the system F^v. $F^v(\alpha^{v,l+1})$ results in Boolean vector comprising only 0-values. It means the fault is detectable on the $(l+1)$-th member. Q.E.D.

The masking property of B-faults depends, first of all on the SSC behavior. When the number of transitions of the SSC is several times more than the number of different output codewords, the masking ability is higher. In this case it is possible that simultaneous activation of several transitions of the SSC corresponding to different states will not damage the SSC output codeword. Since all these transitions produce the same output codeword. Such situation occurs when the same output vectors are produced on transitions from different states as a reaction on the same input vector.

Let us illustrate the masking property using Table 1. We have the input sequence 111, 111, 101, 000, that begins from initial state 1000. Let a fault during the first input 111 changed the next state 0100 for the 1100 but the output 1001010 remained unchanged. In the next clock, this fault has disappeared but the input 111 activates simultaneously 3-th and 5-th transitions instead of 3-th only. Both these transitions have the same output codeword 1001010, and a vector 0110 represents the next state vector. It means that input 101 simultaneously activates 5-th and 6-th transitions instead 5-th transition only. That results in the truth output 1001010 and the same next state 0010.

Now the behavior of the circuit does not differ from the fault free circuit's behavior. The fault has masked itself i.e. the SSC survives.

It is clear that the shorter transient fault then its masking ability is higher. We can increase the masking ability by a special encoding. Let we have an *m-out-of-n* code. The B-fault increases the weight of the proper state code till $(m+k)$. Then C_{m+k}^{m} is the maximal number of states, which can be activated, with the next member of the input sequence γ. This number reaches a minimal value when m is equal to 1.

Decreasing the weight of states codes is another way of increasing the masking ability. We usually assume that the length of the code is as short as possible. Let us disregard this assumption.

If we increase the length of codes preserving their weight (the weight is more than 1), we can also increase the masking ability of B-faults. In fact, when the length of code is more then the length of the minimal code, only some codes from C_{m+k}^{m} codes of SSC states are reachable. It reduces the number of states activated with the next member of the sequence $\tilde{\gamma}$. Therefore, increasing the masking ability requires an additional overhead.

5. Conclusion

We have proposed a technique where a totally self-checking sequential circuit (TSC SSC) reacts differently to appearance of permanent and transient faults. Namely, in the case of transient faults, the TSC SSC is not marked as erroneous immediately, as has been accepted by known practice, but is given "a chance" to survive. The chance is ensured by providing the SSC with a property of partial monotony.

The SSC having this property is able not just to continue its proper functioning after occurrence of the fault, but to become a fault free, i.e. to survive.

References

[1] Diaz, M., and P. Azema, "Unified design of self-checking and fail-safe combinational circuits and sequential machines", IEEE Transaction on Computers, C-28, March 1979, pp. 276-281.

[2] Ozguner, F., "Design of totally-self-checking asynchronous and synchronous sequential machines", Proc. Int. Symp. Fault-Tolerant Computing, 1977, pp. 124-129.

[3] Jha, N.K., and S.J. Wang, "Design and synthesis of self-checking VLSI circuits and systems", IEEE Trans. CAD, 12, no. 6, June 1993, pp. 878-887.

[4] Bolchini, C., R. Montandon, F. Salince, and D. Sciuto, "Design of VHDL-Based Totally Self-Checking Finite State Machines and Data-Path Descriptions", IEEE Transaction on Very Large Scale Integration (VLSI) Systems, Vol. 8, No. 1, 2000.

[5] Levin, I., V. Sinelnikov, "Self-checking of FPGA based Control Units", Proc. of 9th Great Lakes Symposium on VLSI, Ann Arbor, Michigan, 1999, IEEE press, pp. 292-295.

[6] Levin, I., M. Karpovsky, "On-line Self-Checking of Microprogram Control Units", 4-th International On-line Testing Workshop, Capri, 1998, Compendium of papers, pp. 153-159.

[7] Matrosova, A., I. Levin, S. Ostanin, "Self-checking Synchronous FSM Network Design with Low Overhead", Journal of VLSI Design, vol. 11, No1, 2000, pp. 47-58.

[8] Lala, P., "Self-checking and Fault-Tolerant Digital Design", Morgan Kaufmann Publishers, San-Francisco/San-Diego/New-York/Boston/London/Sydney/Tokyo, 2000.

[9] Matrosova A., S. Ostanin., "Self-checking FSM Design with Observing only FSM Outputs", Proc. The 6-th Int. On-Line Testing Workshop, July 2000, pp.153-154.