Self-healing Ability of Sequential Circuits

I.Levin, V.Ostrovsky, S.Ostanin
Tel Aviv University

1. Introduction

Two different ideologies can be considered for handling an error detected in a circuit concurrently with its functioning. The first ideology is based on the immediate marking of the circuit as erroneous when an error is detected. An alternative ideology concentrates on increasing the survivability of the circuit, which means, "to give a chance" to the circuit to continue to work during a number of clock cycles after the error detection, and marking it as erroneous only after a steady error is indicated. Sometimes these several clock cycles are sufficient for the recovery of a transient fault and for returning the circuit to its proper functioning. The first of the above-mentioned ideologies has an advantage in the low fault latency. The second one offers a way for circuits to survive in cases of transient faults.

In the present paper we investigate architectures that enable circuits to survive without introducing any additional overhead. We deal with microcontrollers described by Finite State Machines (FSMs) and implemented as Synchronous Sequential Circuits (SSCs).

We use the on-set realization of output and next state functions. Furthermore, we deal with implementations where both the next state and output equations are unate [1] in state variables and binate in primary input variables. SSCs that are implemented according to such a scheme we call state monotonic SSCs. In most cases using these implementations results in a considerable reduction in overhead [2,3]. Furthermore, and that is substantial in the present work, such SSCs can function properly with the presence of a fault and even recover from the fault. This property is called self-healing.

The proposed synthesis technique implements self-healing SSC and minimizes the required overhead. We investigate the techniques from the point of overheads, assuming that the resulting on-line checking controller will be implemented by a FPGA.

2. Behavior of the SSC in presence of transient faults

To formulate our approach define three modes of functioning of the SSC and describe its behavior in presence of transient faults [1]:

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

S - Silent mode. It is a mode where the presence of a fault does not manifest itself in the form of a non-code output, although the presence of the fault could potentially be detected when the next state lines (or the memory) can be observed. The circuit moves to the silent mode from the fault free mode when a non-code state vector appears on the flip-flop inputs or outputs. From the silent mode the circuit is able either to move to an erroneous mode (**E**), or to revert to the fault free mode.

E - Erroneous mode. It is a mode in which the circuit terminates its proper functioning, i.e. when a non-code output vector has been produced. The circuit is able to move to this mode either from the silent mode or from fault free mode.

The sequential circuit is able to revert to the **F** mode after it's functioning in the **S** mode, which means that the SSC has become fault free again. Such a transition of the SSC from the **S** mode to the **F** mode we call the self-healing. We show that a SSC may have a self-healing property for a given fault and a given input sequence even if it does not have equivalent states. The property of self-healing can be provided for SSCs that are monotonic state variables [2].

3. State Monotonic SSC

The system $\Phi(x_1,...,x_k)=\{\varphi_1(x_1,\ldots,x_k),\varphi_2(x_1,...,x_k),...,\varphi_l(x_1,\ldots,x_k)\}$. is monotonic if, for any pair of Boolean k-tuples A, B such that $A \le B$, the condition $\Phi(A) \le \Phi(B)$ is satisfied. Consider a subset x of Boolean variables, $x \quad \{x_1,\ldots,x_k\}$. The system Φ is partially monotonic in x variables, iff for any pair of Boolean k-tuples A, B $A \le B(x)$ the condition is $\Phi(A) \le \Phi(B)$ satisfied.

Let us consider a specific realization of the SSC. We will call it a state-monotonic SSC. A SSC having a combinational part defined as a system of Boolean functions that are partially monotonic in state variables is defined as a state monotonic SSC.

Obviously, any system of Boolean functions partially monotonic in state variables can be presented in the sum-of-products form, which is unate in state variables.

Consider system Φcorresponding to the combinational part of SSC. Suppose that Φ is partially monotonic in state variables for a state monotonic SSC. Let Φ_f be a system corresponding to the combinational circuit C in presence of a fault *f*. Also suppose that Φ_f is partially monotonic in state variables. If Φimplicates Φ_f, we denote this fault as f_0. If $\Phi_f$implicates Φ, we denote this fault as f_1. Obviously, any unidirectional fault is either f_0 or f_1. It has been proven in [2] that the state monotonic SSC is fault-secure for any transient fault f_0 or f_1.

The above-mentioned feature allows checking only output codewords of the state monotonic SSC, without checking whether transitions of the SSC are correct or incorrect. If a fault leads to an incorrect transition of the SSC, there is no

4. Self-checking SSC architecture

The architecture that supports the self-healing property of the partially monotonic SSCs is a well-known self-checking architecture, that includes the SSC to be checked and an output self-checking checker. Indeed, such architecture detects only output non-codewords. In this case non-code state may appear as memory states and result in self-healing.

Major difficulties in designing self-checking devices are related to the complexity of encoding of check bits and decoding (i.e. verification that a given output is a codeword). We deal with an SSC-based controller where a number of possible output vectors is much smaller than 2^N (where N - is the number of output lines), while the set of possible vectors is known in advance. It was shown that the checker for an SSC controller could be efficiently implemented in the form of "sum-of-minterms" (SOM) for output functions of the controller. An unordered code for output vectors is used since, for these codes, no unidirectional error can transform one codeword into another codeword The main objective in the construction of unordered codes (Berger[4], Smith[5] codes) is to achieve the required properties by using a minimal number of redundant bits. Nevertheless, decrease of the number of bits in many cases does not lead to an expected reduction in the checker's complexity. Thus, an approach that takes into a account checker's complexity at the earlier stages of the code construction becomes very impotent. Moreover, the basis of implementation has to be taken into consideration. Based on these requirements we propose a novel so-called reduced m-out-of-n code. The reduced m-out-of-n code differs from the standard code by having the following two features:

this code is a systematic one;

this code enables dividing the codeword into m fields in such a way that any acceptable codeword has exactly one bit that is equal to one in each of the fields.

The proposed reduced code allows checking each of the fields separately and therefore simplifies the checker.

Introducing of a reduced m-out-of-n code defines a ratio of compatibility/ incompatibility on the set of acceptable output codewords O for a given output code, as follows. Two variables Z_i and Z_j are compatible iff they are both equal to "one" for at least one codeword $o_r \quad O$, otherwise Z_i and Z_j are incompatible.

Subset $T_j \quad Z \left(Z = \{ Z_1, \ldots, Z_{N_z} \} \right)$ is a set of compatible (incompatible) output variables iff any two variables that belong to the set are compatible (incompatible). Obviously, if the set T_i is the set of incompatible output variables, then every acceptable output codeword may contain at most one variable from the set T_i.

Let set Z is divided into subsets $T_1,\ldots,T_m$ of incompatible variables $Z = T_1 \quad \ldots \quad T_m$, $T_i \cap T_j =$, for any $i \neq j$.For constructing the reduced m-out-of-n code we put each subset T_i into the one-to-one correspondence to a specific check bit c_i; c_i=1 only in those output codewords that have no output variables from T_i.

Then every output word from the proposed code contains exactly m bits that are equal to one, according to the number of subsets T_i.

We estimate the checker's complexity by the number L of 4-input LUTs that have been used for implementing the checker. Let N output signals be partitioned into m subsets for coding. Each of these subsets includes N_i elements ($\sum_{i=1}^{m} N_i = N$). In this case: $L = 2\left(\sum_{i=1}^{m} \lceil (N_i - 1)/2 \rceil + m - 1\right)$, where $\lceil X \rceil$ is least integer upper bound of X. Thus we have the following bounds for complexities of checkers: $(N + m - 2) \le L \le (N + 2m - 2)$.

5. Experimental Results and Conclusions

We investigated the self-healing ability of SSCs on a number of benchmarks [6] in presence of transient faults. Experimental results show that the newly proposed technique for designing totally self-checking SSCs based on the reduced m-out-of-n coding results in a hardware overhead of about 30%, while the Berger coding implementations require an overhead of about 50% and the Smith coding – about 40%. In other words, the authors' approach leads to a reduction of about 10-20% of the overhead as compared with the traditional methods.

The described above self-healing property characterizes a plurality of MCNC benchmarks. The average number of clocks required for SSC recovering is about 1-2 clocks. The self-healing time is approximately equal to the latency. It means that a future of the SSC (to recover or to "die") depends of the circuit's behavior during several clocks after moving to the silent mode. The average percent of sequences on which SSC benchmarks is recovered is about 20%. This parameter looks promising and motivating for the future investigation of the self-healing.

It should be noted that the SSC scheme does not require performing in it any changes for becoming a self-healing scheme. Our research has shown that the self-healing property is intrinsic to a great number of SSC schemes, even to the majority of them. The ability of recovering is a feature of the scheme, which, as well as other its features, depends on the functional description of the scheme and on the manner of its implementation.

Intuitively, the first opinion on the possibility of the SSC self-recovering (if performed without storing the internal states at specific characteristic points for further returning to such states if a fault occurs) is such that it appears only if so-called equivalent states exist in the SSC. In practice, however, the above –mentioned intuitive opinion appears to be erroneous, and the self-healing just proves to represent a subject of more detailed and deeper investigation.

Indeed, let an automaton, due to a fault, performed an erroneous transition to an internal state q(i) instead of transiting to an internal state q(j). Let this faulty transition remained unnoticed by the output control, namely one and the same

The equivalency of the internal states is a sufficient condition for the recovery, though it is not a necessary one.
Let's call the states q(i) and q(j) similar if there exists at least one input sequence of codewords, in response to which the automaton scheme generates one and the same output sequence and transits to one and the same internal state q(k). It should be noted that one of the similar states (in our example, q(i)) may be absent in the original description of the automaton and appears only when a particular fault occurs. Obviously, the automaton will keep the self-recovering (self-healing) ability iff the erroneous state q(i) and the correct state q(j) are similar.
The condition of similarity of states is less strict than the condition of the states equivalency, since in the case of similarity it is sufficient that the automaton's reaction be the same only for at least one input sequence, while in the case of equivalency the automaton's reaction must be identical to any input sequence. It is interesting to note that existence of such similar states in an automaton does not obligatory lead to any redundancy in its defined structure. Analysis has been performed for the benchmarks automata, and has shown that most of them comprise such similar states and, therefore, possess the self-healing ability.

References

[1] Levin I., Ostrovsky V., Ostanin S., Karpovsky M. "Self-checking Sequential Circuits with Self-healing Ability", Proceedings of the 12-th ACM Great Lake Symposium on VLSI, New York, April 2002, pp. 71-76.
[2] 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.
[3] Lala, P., "Self-checking and Fault-Tolerant Digital Design", Morgan Kaufmann Publishers, San-Francisco / San-Diego / New-York/ Boston/ London/ Sydney/ Tokyo, 2000.
[4] Berger, J. M., "A Note on Error Detection Codes for Asymmetric Binary Channels", Inform. Contr., Vol. 4, March 1961, pp. 68-73.
[5] Smith, J., "On Separable Unordered Codes", IEEE Transaction on Computers, Vol. C-33, No. 8, August 1984, pp. 741-743.
[6] Lisanke R., "Logic synthesis benchmark circuits", International Workshop on Logic Synthesis, Research Triangle Park, NC, May 1989.