Optimization of error detecting codes for the detection of crosstalk originated errors

M. Favalli
DI - University of Ferrara
Viale Saragat, 1 - 44100 Ferrara, Italy

C. Metra
DEIS - University of Bologna
Viale Risorgimento, 2 - 40136 Bologna, Italy

Abstract

This work applies weight based codes [1] to the detection of crosstalk originated errors. This kind of faults, whose importance grows with device scaling, may originate errors that are undetectable by the mostly used error detecting codes in VLSI ICs. Conversely, such errors can be easily detected by weight based codes that, however, have smaller encoding capabilities. In order to reduce the cost of these codes, a graph theoretic optimization is used. Moreover, new applications of these codes are explored regarding the synthesis of self-checking FSMs, and the detection of errors related to the clock distribution network.

1. Introduction

Self-checking circuits are one of the basic building blocks of fault tolerant systems. Therefore, they are widely used in the traditional domains of safety critical computation. Their application to consumer electronics, instead, is now motivated by the growing reliability requirements of this kind of applications that contrast the reliability problems of deep submicron technologies [2]. In such cases, self-checking circuits provide a viable approach to fault tolerance because of their low cost with respect to modular redundancy systems.

A self-checking (SC) circuit is essentially based on a functional unit (FU) that provides an information flow protected by means of an error detecting code, and of a checker (C) that continuously verifies the correctness of such information. This allows the detection of errors as soon they occur, thus avoiding their propagation through the whole system. In addition, error indications can be used to enable the activation of error recovery procedures operating at higher levels such as software ones.

In SC systems, the choice of the error detecting code is mainly based on its effectiveness and on design cost considerations which regards both the FU and the C implementations. In the case of CMOS VLSI circuits, the most common kind of codes ranges from the parity one, up to duplication (or two-rail implementation). As intermediate solutions, unidirectional error detecting codes [3] are often used because, so far, this kind of errors was recognized as the most common in digital systems. These codes include the m-out-of-n and the Berger codes that have been extensively used in the design of SC systems [4, 2].

For such a kind of systems, the totally self-checking (TSC) goal is the main property ensuring reliable operations in the presence of faults. In particular, a circuit achieves the TSC goal with respect to a class of faults, if the first erroneous output due to a fault results in an error indication [5].

In order to provide design rules for SC circuits achieving the TSC goal, more detailed properties have been defined for both functional units and checkers.

A circuit is fault-secure (FS) 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 (ST) 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 a circuit is both FS and ST it is said to be totally self-checking (TSC) [5].

A wider class of circuits achieving the TSC goal is given by those satisfying the strongly fault-secure (SFS) property. A circuit is said to be SFS [6] with respect to a set of faults F if for every fault in F, either: a) the circuit is ST, or b) the circuit is FS, and, if another fault from F occurs in the circuit, then either (a) or (b) is true for the fault sequence.

As can be seen from the above definitions, the correct operations of a SC circuit are strongly related to the considered set of faults. As this set represents the main characteristic of the possible failures in current technologies, as the reliability of SC circuit operations is ensured with a better confidence.

Traditionally, the stuck-at fault model has been mostly used as the main reference in the design of SC circuits.

Under the stuck-at fault assumption, design rules for circuits achieving the TSC goal have been presented [7]-[16]. Of course, it is well known that the stuck-at fault model cannot represent all the possible failures in the current technology. As a consequence, design rules have been derived for bridging and delay faults [17, 18].

More recently, the problems of SC circuits in the presence of crosstalk faults (that represent a problem of growing im-

portance with ICs scaling [19]) have been discussed in [20] and in [21]. In particular, it is shown that crosstalks may inherently give rise to bidirectional errors[1] that are not detectable by typical error detecting codes used in SC systems. The problem can be solved by using ad-hoc on-line detection circuitry [21], that, however, may be not available in some design style.

In addition to such a problem, even when simply considering the stuck-at fault model, the TSC or the SFS properties pose some constraint on the design of functional units. In case different outputs contain the same subexpression in true and complemented form, such constraints prevent from the full exploitation of the fan-out. In such a case, the whole cone of logic generating the common expression may need to be duplicated, thus leading in some cases to large overhead costs [16].

To approach the design of on-line testable functional units in a cost effective way, in [1] the use of weight-based codes has been proposed by Das and Touba. Such codes have a weight (possibly different from 1) assigned to each information bit, while the check bits provide an encoding of the sum of the weights of all the information bits. These codes have very interesting error detection capabilities since they can detect some of the bidirectional errors affecting two signals and all the unidirectional errors. In [1], they have been applied to the on-line testing of functional combinational blocks, and they have been shown to achieve very high fault-coverages. This target is achieved with respect to stuck-at faults, with no modification to the functional unit (a part from the need to add the checkbit generator). From this point of view they compare favorably to Berger codes. The FU, however, does not achieve the TSC goal because of the undetected faults.

In this work, we propose an approach that exploits the properties of this class of codes in order to solve some of the problems related to crosstalk faults and to the synthesis of TSC functional units. In particular, we provide a weight selection strategy to find a code that is capable to detect the most likely bidirectional errors and all the unidirectional ones, while presenting a limited degradation of the code efficiency with respect to the case of the mostly used error detecting codes for VLSI SC circuits.

In particular, an optimization strategy is proposed that exploits information about the layout (in the case of crosstalks) or the logic structure (in case of functional units) of the circuit. Based on such a knowledge, the most likely pairs of signals affected by bidirectional errors can be individuated and annotated in a graph. Such a graph is then used to find a suitable low cost code capable to detect them.

In the specific case of crosstalks, the problem is exactly solved as a graph coloring problem.

[1]This term is here used to indicate errors affecting either bits at logic 1 or 0 within the same word.

In the case of functional units, weight-based codes are used to reduce the additional logic needed to achieve the TSC goal. Notice that this use is dual to that proposed in [1], where the logic is fixed to its minimum, and weights are selected to maximize coverage. Here, instead, the coverage is fixed at its maximum, while we attempt to minimize the required logic. This target is performed by means of a simple heuristic that has been verified to provide some area saving in the specific cases of SC FSMs.

In [1], checkbits are added to existing information bits, thus, in practice, extending the Berger codes. The considered cases, instead, do not provide starting information bits, and we make a different use of weight-based codes that can be viewed as an extension of the *m*-out-of-*n* ones.

Finally, the use of weight based codes has been applied to the detection of errors related to the clock distribution tree in high speed circuits. In fact, we show that a delay fault affecting a subtree of such a network may give rise to bidirectional errors.

2. Weight-based codes

The kind of codes proposed in [1] is a generalization of the Berger code that is known to be the more efficient all unidirectional error detecting code.

In the Berger code, codewords are composed by I information bits and C check bits ($C = \lceil \log_2 I \rceil$). The C check bits are given by the 1-complement to the binary encoding of the 1s number in the information bits in the B1 encoding scheme, while they provide the binary encoding of the number of 0s in the B0 encoding scheme. In practice, the number of ones can be expressed as $k = \sum_I w_i v_i$, where $w_i = 1 \forall i \in I$ and v_i is the logic value of the i-th information bit.

Weight-based codes are obtained by assigning different weights $w_i \in \mathrm{N}$ to the information bits. The checkbits are assigned in the following way: if $k = \sum_I w_i v_i$ for a given word, then the C checkbits may be calculated as: i) the 1 complement to the binary encoding of k (B1 extension); ii) the binary encoding of $\sum_I w_i - k = \sum_I w_i v_i'$ (B0 extension).

This kind of code detects all unidirectional errors and the bidirectional errors affecting the information bits that result in a variation of the actual value of k. For instance, a single (i.e. affecting two bits) bidirectional error is detected if the weight of the two affected bits is different. Both the error detection capabilities and the code efficiency depend on the set of used weights $\mathcal{W} \subset \mathrm{N}$. In general, in weight-based Berger like codes (thereafter denoted as B^*), C is larger than in the Berger code case. For instance, if $\mathcal{W} = \{1, 2, ..., I\}$, C is equal to two times the number of check bits required by a conventional Berger code, and the weight-based code can detect all single bidirectional errors and all unidirectional errors.

The B^* code has a lower efficiency (i.e. number of encodable informations with respect to the codeword size) than the

corresponding Berger codes. In fact, it must be $2^C \geq \sum_I w_i$. It is, therefore, important to minimize the cardinality of such a set on the basis of the occurrence probability of the bidirectional errors.

Also constant weight unordered codes such as the *m*-out-of-*n* code can be generalized in the same way, even if this case is not considered in [1]. In this case, the resulting codes are not separable and cannot be used to add checkbits to existing functions.

All words (c) belonging to the *m*-out-of-*n* code satisfy the relationship:

$$W(c) = \sum_{i=1}^{n} w_i v_i = m \ ,$$

where $w_i = 1 \forall i$ and v_i is the value of the i-th ($i = \{1, 2, ..., n\}$) bit of the considered word.

Such codes can be generalized as:

$$W(c) = \sum_{i=1}^{n} w_i v_i = m \ ,$$

where $w_i \in \mathrm{N}$ is the weight of the i-th bit. Notice that now m may be larger than n. Let also $\mathcal{W} \subset \mathrm{N}$ be the set of used weights, and let $\mathcal{B}(w)$ be the set of indexes i such as $w_i = w$.

This kind of code (that will be thereafter denoted as CW^* to remark that it is an extension of a constant weight error detecting code) is still able to detect all the possible unidirectional errors. Conversely, it can detect all single bidirectional errors involving a pair of bits i and j such as $w_i \neq w_j$. In fact, in the case of a bidirectional error involving such lines, it is $W(c) = m \pm |w_i - w_j|$, so that a checker can detect the error.

In order to keep the complexity of checkers as low as possible, in the remainder of this work we will consider only the cases with $w_i \leq n$.

If all w_i differ to each other, it is evident that all single bidirectional errors can be detected. In the considered cases, this latter constraint is satisfied if $w_i = i$. As it will be shown later in this paper, several VLSI applications do not require the detection of all single bidirectional errors, so that a smaller set of weights can be used.

Of course, also the CW^* code has in general a lower efficiency than the corresponding *m*-out-of-*n* codes. In particular, for large values of n (i.e. $n \gg max\{w_i\}$), an upper bound to the CW^* encoding capabilities can be easily obtained by considering a *m*-out-of-*n* code with $n = \sum_i w_i$, m equal to that of the CW^* code, and by restricting it to the codewords corresponding to those of the weight based code.

Such restricted *m*-out-of-*n* code can be obtained by expanding each bit i of the weight based code into w_i bits, and by allowing these bit to assume only the configurations of all 1s or all 0s. For instance, consider a CW^* code where $m = 3$, $w_i = i$, $n = 3$, the codeword 110 is transformed into 111 000. If n is large, these w_i bits would have an almost uniform distribution in a conventional *m*-out-of-*n* code. Therefore, the ratio between the number of configurations of the *m*-out-of-*n* code and of the restricted code obtained by fixing w_i bits is $2/2^{w_i}$. By repeating this reasoning for all the bits, the lost of encoding information in the restricted code is proportional to:

$$\rho = \prod_i 2^{w_i - 1} \ .$$

Since the number of configurations of the *m*-out-of-*n* code is:

$$\binom{n}{m} = \binom{\sum_i w_i}{m} \ ,$$

the number (e) of configurations of the weight based code is given approximately by:

$$e = \frac{\binom{\sum_i w_i}{m}}{\prod_i 2^{w_i - 1}} \ .$$

Such a value is an upper bound because discrete constraints typically reduce the number of possible configurations that is typically contained in the interval $[e/2, e]$.

The maximal efficiency of the CW^* code for a given $\mathcal{W}$ is obtained when it is $m = \lceil 1/2 \sum w_i \rceil$ or $m = \lfloor 1/2 \sum w_i \rfloor$. When this condition is verified, of course, the code efficiency is as larger as the cardinality of $\mathcal{W}$ (that in this way approaches the optimal case where $w_i = w_0 \ \forall i$) it is smaller. It is, therefore, important to minimize it on the basis of the occurrence probability of the bidirectional errors.

3. Crosstalk faults

In the case of crosstalks, in [20] it has been shown that some of the most frequently used error detecting codes in SC circuits may be unable to detect the resulting errors. This problem has been shown to be worse with the *m*-out-of-*n* code case with respect to other codes. The problem can be solved by using the checking circuitry presented in [21].

If such a detector is not available, the problem can be avoided by the use of weight based codes. In particular, from the layout structure, some inference can be made on the probability of occurrence of a crosstalk fault. At this regard, we use the same hypotheses of [20]:

- the effects of crosstalks on ICs timing under nominal conditions are accounted by timing analysis;
- faults may still occur because, for instance, of lack of insulating material or of wrong mask alignments.

Therefore, we can consider single crosstalks with a good approximation.

Consider, for instance, the possible layout structure of a bus as it is represented in Fig. 1. In such a case, it is rather evident that the probability of a crosstalk between two adjacent lines or two overlapping lines is much higher than that between other lines. This kind of conditions can be well represented by an undirected graph with a vertex for each signal

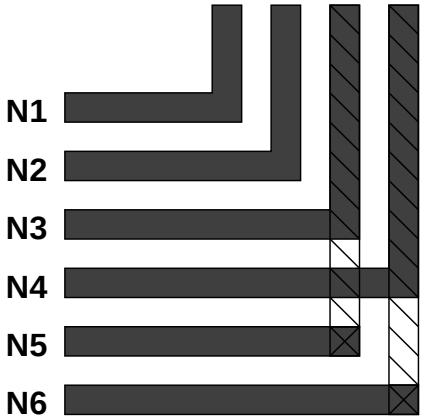

Figure 1. Example of a possible bus structure. The bus makes use of two metal lines. Crosstalks are considered possible between adjacent or overlapped lines.

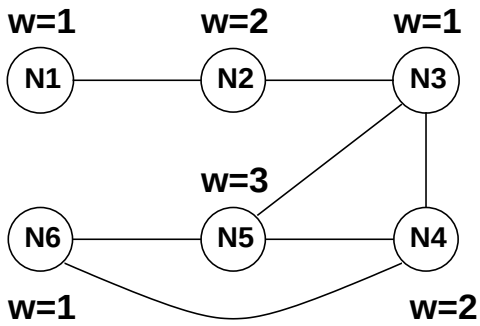

Figure 2. Graph representing the possible crosstalks in the bus layout structure illustrated in Fig. 1. The nodes correspond to signal wires, while an edge is present if crosstalks between the two wires are possible. The figure shows also a possible choice of weights (i.e. coloring) that makes detectable such crosstalks.

possibly affected by a crosstalk, and an edge between two vertex if a crosstalk is possible between the two corresponding signals. Therefore, an edge represents a possible single bidirectional error. The graph related to the example of Fig. 1 is shown in Fig. 2. This graph can be used to determine, for a given layout structure, the optimal kind of code capable to detect the bidirectional errors due to crosstalks. Such errors are represented by edges and they are detectable only if two directly connected vertex have different weights.

The problem of assigning weights to the signals in the bus is, therefore, equivalent to the well known one of graph coloring. Fig. 2, for instance, shows a possible coloring of the graph representing the possible crosstalk problems in the layout of Fig. 1.

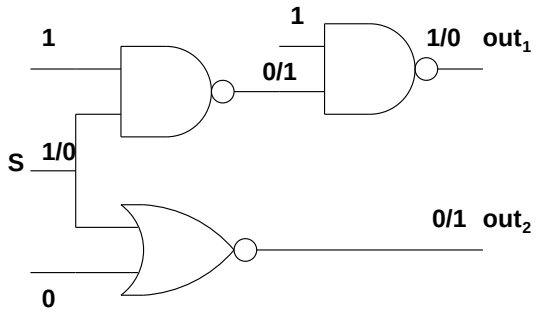

Figure 3. Example of the problems encountered in the design of self-checking functional units in case of signal reconvergency.

3.1. Application to the case of decoder outputs

As an application, consider the outputs of a decoder used to select devices. Such outputs provide a 1-out-of-n code that is capable to detect all the possible unidirectional errors, but not the possible single bidirectional errors coming from crosstalk faults. To detect such errors the approach outlined above can be used. In particular, in case the decoder output wires ($d_i,\ i = 1,....,n$) present a parallel layout structure, the resulting graph would be linear (i.e. $\forall i \neq 1,n,\ e_{i,j} = 1$ only if $j = i \pm 1$), so that it can be colored by using only two colors. Therefore, it is $\mathcal{W} = \{1,2\}$, thus requiring an additional bit with weight 1 that is 0 when the decoder output at the logic 1 has weight 1, and 0 when the decoder output at logic 1 has weight 2. Hence, for all codewords, it is $W(c) = 2$. In the presence of a bidirectional error affecting two decoder outputs, the weight changes from 2 to 1 or 3. If the crosstalk affects the n-th decoder output and the additional checkbit, any bidirectional error is still detectable if the n-th bit has weight 2, because the additional checkbit has weight 1.

4. Synthesis of SC FSMs

Fig. 3 instantiates the problem in the synthesis of SC functional units based on unidirectional error detecting codes. Two outputs share a common fan-in cone (whose output is represented by the signal s), but the output signal of such a cone reaches the two outputs with paths presenting a different parity in the number of inversions (polarity). In case of an fault on s both the outputs may be affected by an error. This requires to duplicate the considered cone of logic. Even if more sophisticated approaches to the design of SC functional units exist based also on boolean considerations [16], this problem may easily lead to overheads similar or larger than duplication. It is obvious, that, if it is possible to assign two different weights to the two output signals, the cone must no longer be duplicated.

We will use this idea in order to reduce the overhead of the combinational part of a SC FSM. In such a case, however,

the problem of finding a suitable CW^* code is much more complex than in the case of crosstalks. In fact, it depends not only the choice of the set of codewords, but also on the way in which they are related to states.

Hence, we do not propose a general approach to the problem, but we illustrate a simple procedure that, in the specific case of the synthesis of SC FSMs, illustrates the possible advantages of weight based codes.

Notice that the design of SC FSMs presents also several problems [22] which are not considered here. With respect to them, in the considered work, we suppose to check present state variables (i.e. register outputs). This solves the problem of don't care states without the need to add code-disjoint features to the combinational part of the machine.

In particular, we start from a STG, and we implement a SC synchronous circuit. The first step to be performed is state encoding. In our simple strategy, we make the arbitrary choice to use $\mathcal{W} = \{1,2\}$ and $\mathcal{B}(1) = \mathcal{B}(2) \pm 1$. In this way the degrees of freedom that are allowed by the code choice are not saturated and will be subject of further research. The advantage of such a choice is essentially due to the fact that, with respect to the use of *m*-out-of-*n* codes, we need a number of bits that is equal to n or $n+1$, where n is the minimal number of bits required by the most efficient *m*-out-of-*n* code to encode all the states.

Once the states have been encoded, the network is synthesized by means of SIS [23] simply targeting area reduction. Of course, this circuit is not TSC. Hence, it is traversed counting the duplications required in order to avoid that any gate reaches two outputs characterized by the same weight in the selected code with different polarities. In this way, a cost estimate of the SC implementation of the circuit is achieved. While performing such traversal, a coefficient $a_{i,j}$ is evaluated for each pair of outputs i and j. It is initialized to 0, and it is incremented by 1 for any gate duplication required by a signal reaching i and j with a different polarity.

The coefficients $a_{i,j}$ are then used to refine the code selection. In fact, consider the case where a pair of outputs (i, j) has a very low value of $a_{i,j}$ and $w_i \neq w_j$, while another pair (k, l) has a high value of $a_{k,l}$ (i.e. the two outputs share a large number of gates with different polarities), but $w_k = w_l$. In such a case, it is evident that it would be convenient if the relationships between the weights of such signals are inverted in order to have $w_i = w_j$ (because this implies a small number of duplications), and $w_k \neq w_l$ (to avoid the large number of duplications related to such signals).

In the practice, coefficients $a_{i,j}$ can be interpreted as weighted edges of a graph (with the FU outputs as vertex) providing similar constraints to that used in the case of crosstalks. Such a graph can be used to refine the state encoding by using a heuristic searching for a weight reassignment minimizing the cost function given by the sum of all $a_{i,j}$-s between nodes with the same weight (in practice, this is a max cut partitioning problem).

Notice that, for certain number of states in the original STG, the use of CW^* codes does not require additional state vars with respect to the *m*-out-of-*n* case and, therefore, it would, in general, be a convenient choice.

4.1. Results

By using this heuristic, we have considered some sequential benchmark (state machines) from the *mcnc* set [24]. The STG has been minimized by means of the program *stamina* (that is included in SIS) and states have been encoded by using suitable *m*-out-of-*n* (those featuring the $m = \lceil n/2 \rceil$, or $m = \lfloor n/2 \rfloor$) and CW^* codes. The CW^* code initially is simply randomly assigned. The combinational part of such sequential machines corresponding to the next state function has been optimized by means of SIS and mapped on a standard logic library. Then both the versions have been made TSC with respect to stuck-at faults by means of topological analysis and duplication.

The achieved results together with some statistics on benchmarks are shown in Tab. 1. In particular, the table shows the number of state variables used in a *m*-out-of-*n* and in a CW^* encoding, and, for both the kind of encodings, it gives the number of gates in the area optimized version. Finally, results are shown for the SC versions by providing the number of gates for both the encodings and the relative saving in the number of gates.

As can be seen, in 8 out of 10 cases the approach based on weight based codes works better than the *m*-out-of-*n* case (even when requiring an additional state variable) with a relative saving of the 31%. For the circuits providing unsatisfactory results (namely **ex4** and **dk16**), a second iteration on code selection has been performed resulting in savings of 5% and 1%. It should be noticed that also in these unfavorable cases, the results achieved are comparable to those obtained by the *m*-out-of-*n* code.

This result is promising because the state encoding has been performed in a random way, and further savings may be possible by merging the use of weight based codes with state assignment procedures. Let us also notice that we considered only the gate number, since we performed only topological transformations. The savings slightly increase (33%) when considering the literal number.

5. Application to the detection of faults affecting the clock distribution tree

In order to show that weight based codes may be applied to a large set of possible problems, here a simple application to the case of high speed integrated circuits is proposed. In these circuits, the clock distribution network is a critical design aspect. In particular, the relevant resistive effects affecting interconnections in submicron ICs, the large geometrical extension, the large fan-out of the clock distribution network require an extensive use of buffering.

bench.	inputs	states	state vars.		gates (AO)		gates (SC)		savings
			m-out-of-*n*	CW^*	*m*-out-of-*n*	CW^*	*m*-out-of-*n*	CW^*	
lion	2	4	4	4	45	45	75	69	8%
mark1	12	5	6	6	102	102	257	118	117.7%
ex4	5	14	6	7	82	86	161	190	-16%
ex6	5	8	5	6	80	84	320	161	98%
tbk	6	16	6	7	428	431	1281	1176	8%
ex2	2	14	6	7	118	131	316	297	6%
scf	27	97	9	9	751	644	2522	1985	27%
cse	7	16	6	7	267	333	989	761	30%
dk16	2	27	7	8	302	284	924	970	-5%
s1	8	20	6	7	301	331	933	677	37%

Table 1. Results achieved by the proposed state encoding technique for a set of FSM STGs benchmarks.

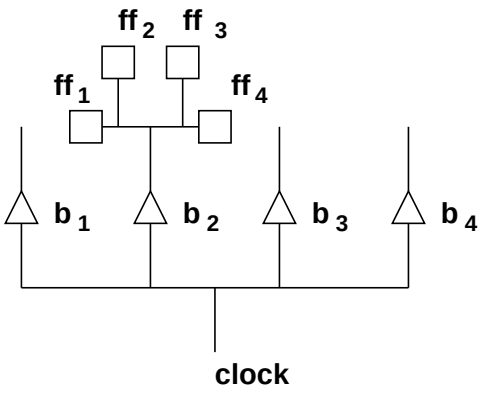

Figure 4. Example of clock distribution network.

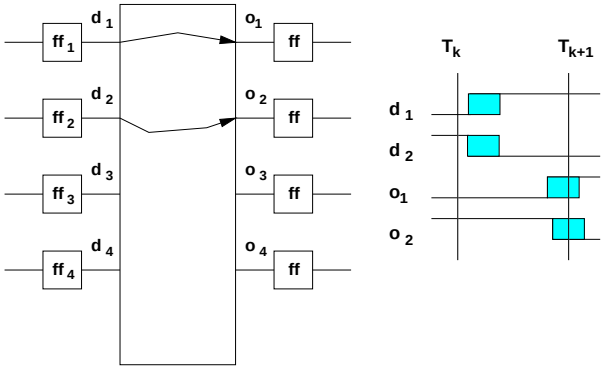

Figure 5. Example of bidirectional error generated by two flip-flops controlled by a delayed clock signal.

The reliability of such network is fundamental for the whole IC's behavior. In particular, when the clock speed grows it may become sensitive to faults such as delay and transient faults affecting the buffers, or crosstalks. From the point of view of delay faults, classical approaches usually consider defects as to be concentrated inside the combinational part of the network. Under such hypothesis, FUs can be designed that achieve the TSC goal with respect to single path or gate delay faults by using unidirectional error detecting codes [18].

Unfortunately, in case of delay faults affecting the clock circuitry, bidirectional errors may be in order. Consider, for instance, a delay fault affecting the buffer b_2 in the clock distribution network illustrated in Fig. 4. Because of such a fault, the driven flip-flops will sample their input values with a delay. Suppose that in the fault-free circuit, the current state of ff_1 is 0 and the next state is 1, while, the current state of ff_2 is 1 and its next state is 0. In the faulty circuit, one of these behaviors may occur: 1) the flip-flop inputs have started to switch when the delayed sampling occurs (notice that this may occur if the combinational network providing such inputs is fast); 2) the correct inputs are sampled but the flip-flops outputs are delayed (Fig. 5). It is evident that in case (1) a bidirectional error may occur. In case (2), the logic fed by the flip-flops' output signals is potentially affected by an input bidirectional error. If such signals (d_1 and d_2) are directly checked by an unidirectional error checker, depending on the delay defect size, the checker may not provide an error indication. On the other hand, if such signals feed a functional unit, even if it has been designed to be TSC, depending on the additional delays and the observability of the affected signals, such a fault may result in output delayed transitions. Therefore, it may give rise to the sampling of a bidirectional error (Fig. 5). The only solutions proposed so far to deal with problems affecting the clock distribution network require the use of additional checking circuitry [25, 26]. In case this is not available, the circuit would not achieve the TSC goal.

As an alternative, the use of weight based codes may approach this problem by assigning different weights to the bits sampled by flip-flops whose clock input signal is fed by the same buffer. Also in this case, the size of $\mathcal{W}$ can be reduced by analyzing the network fed by the considered cluster of flip-flops. In particular, timing analysis allows to determine those flip-flops feeding paths characterized by an high prop-

agation delay. These are the most sensitive to delays in the clock circuitry, and require the use of different weights.

6. Conclusions

In this work, the use of weight based codes that have been introduced in [1] for the on-line testing of combinational blocks, has been applied to the cases of: a) the detection of crosstalk faults; b) the design of self-checking finite state machines; c) the detection of errors originated by faults affecting the clock distribution network.

Differently from [1], the used weight based codes are an extension of the *m*-out-of-*n* code. For such codes optimizations strategies are presented based on the likely of occurrence of errors.

In cases, a) and c) errors undetectable by other kind of codes are made detectable, while in case b) the achieved results show an average reduction in the number of gates equal to 31%.

References

[1] D. Das and N. Touba, "Weight-based codes and their application to concurrent error detection of multilevel circuits," in *Proc. of IEEE VLSI Test Symp.*, pp. 370 – 376, 1999.

[2] M. Nicolaidis and Y. Zorian, "On-line testing for VLSI - a compendium of approaches," *J. of Electronic Testing: Theory and Application (JETTA)*, vol. 12, pp. 7 – 20, 1998.

[3] S. Piestrak, "Design of self-testing checkers for unidirectional error detecting codes," in *Monographs No. 24, Oficyna Wyd. Polit. Wrocl., Wroclaw*, 1995.

[4] N. Jha and S. Kundu, *Testing and Reliable Design of CMOS Circuits*. Kluwer Academic Publishers, 1990.

[5] D. A. Anderson and G. Metze, "Design of Totally Self-Checking Circuits for m-out-of-n Codes," *IEEE Trans. on Computers*, vol. C-22, pp. 263 – 269, Mar. 1973.

[6] J. E. Smith and G. Metze, "Strongly Fault Secure Logic Networks," *IEEE Trans. on Computers*, vol. C-27, pp. 491 – 499, June 1978.

[7] M. Diaz, P. Azema, and J. M. Ayache, "Unified Design of Self-Checking and Fail-Safe Combinational Circuits and Sequential Machines," *IEEE Trans. Comput.*, vol. C-28, pp. 276 – 281, March 1979.

[8] D. Sciuto, F. Salice and M. Sami, "Synthesis of multi-level self-checking logic," in *Proc. of IEEE Int. Work. on Defect and Fault Tolerance in VLSI Systems*, pp. 115 – 123, 1994.

[9] N. K. Jha and S.-J. Wang, "Design and Synthesis of Self-Checking VLSI Circuits," *IEEE Trans. on CAD*, vol. 12, pp. 878 – 887, June 1993.

[10] T. Nanya and M. Uchida, "The Design of Strongly Fault-Secure and Strongly Code-Disjoint Combinational Circuits," in *Proc. of Joint Fault-Tolerant Computing Symposium*, pp. 245 – 250, 1989.

[11] V. Saposhnikov, A. Dmitriev, M. Goessel, and V. Saposhnikov, "Self-dual parity checking - a new method for on-line testing," in *Proc. of IEEE VLSI Test Symp.*, pp. 162 – 168, 1996.

[12] K. De, C. Natarajan, D. Nair, and P. Banerjee, "RSYN: asystem for automated synthesis of reliable multilevel circuits," *IEEE Trans. on VLSI*, vol. 2, no. 2, pp. 186 – 195, 1994.

[13] F. Busaba and P. Lala, "Self-checking combinational circuit design for single and unidirectional multibit error," *J. of Electronic Testing Theory and Application*, no. 5, pp. 19 – 28, 1994.

[14] N. A. Touba and E. J. McCluskey, "Logic synthesis techniques for reduced area implementation of multilevel circuits with concurrent error detection," in *Proc. of IEEE Int. Conf. On Computer Aided Design*, pp. 651 – 654, 1994.

[15] C. Bolchini, F. Salice, and D. Sciuto, "A novel methodology for designing TSC combinational networks based on the parity bit code," in *Proc. of IEEE Eur. Design and Test Conf.*, pp. 440 – 444, 1997.

[16] C. Bolchini, F. Salice, and D. Sciuto, "Designing networks with error detectio properties through the fault error relation," in *Symp. on Defect and Fault Tolerance in VLSI Systems*, pp. 290 – 297, 1997.

[17] M. Nicolaidis, "Shorts in Self-Checking Circuits," *J. of Electronic Testing: Theory and Application*, vol. 1, pp. 257 – 273, 1991.

[18] C. Metra, M. Favalli, and B. Riccò, "On-line detection of bridging and delay faults in functional blocks of CMOS self-checking circuits," *IEEE Trans. on CAD*, vol. 5, no. 4, pp. 770 – 776, 1997.

[19] D. Sylvester and K. Keutzer, "Getting the bottom of deep submicron," in *Proc. of IEEE Int. Conf. On Computer Aided Design*, pp. 203 – 211, 1998.

[20] M. Favalli and C. Metra, "Bus crosstalk fault error detection capabilities of error detecting codes for on-line testing," *IEEE Trans. on VLSI Systems*, vol. 3, no. 7, pp. 392 – 396, 1999.

[21] C. Metra, M. Favalli, and B. Riccó, "Self-checking detection and diagnosis for transient, delay and crosstalk faults affecting bus lines," *IEEE Trans. on Computers*, vol. 49, no. 6, pp. 560 – 574, 2000.

[22] S. Piestrak, "Limitations of design methods of self-checking synchronous sequential machines," in *fast-abstract presented to FTCS*, 1999.

[23] E. M. Sentovich et al., "SIS: A System for Sequential Circuit Synthesis," tech. rep., University of California, Berkeley, 1992.

[24] R. Lisanke, "Logic synthesis and optimization benchmarks user guide v. 2.0 - technical report." Microelectronics Center of North Carolina.

[25] M. Favalli and C. Metra, "Testing scheme for IC's clock," in *IEEE European Design and Test Conference*, pp. 445 – 449, 1997.

[26] C. Metra, M. Favalli, and B. Riccò, "Concurrent checking of clock signal correctness," *IEEE Design & Test*, pp. 42 – 48, Oct. 1998.