DESIGN OF SUFFICIENTLY STRONGLY SELF-CHECKING EMBEDDED CHECKERS FOR SYSTEMATIC AND SEPARABLE CODES

Niraj K. Jha
Department of Electrical Engineering
Princeton University
Princeton, NJ 08544

ABSTRACT

Checkers are circuits that are used to monitor the encoded outputs of functional circuits. The concept of totally self-checking (TSC) circuits is well-known. If the first erroneous output is a non-codeword then the TSC goal is said to be achieved. We have recently defined a new class of checkers which meet the TSC goal, namely, the strongly self-checking (SSC) checkers.

Systematic and separable codes are frequently used for error-detection in VLSI circuits. All the possible information symbols are assumed to be present in systematic codes, but not in separable codes.

A TSC or SSC checker can be guaranteed to meet the TSC goal only if they receive the required set of codeword tests from the functional circuit. However, in practice, it is not always possible to ensure this for embedded checkers. In this paper we present a design for a sufficiently strongly self-checking (SSSC) embedded checker for systematic and separable codes. SSCS checkers can be used when it is impossible to design a TSC or SSC checker based on the available set of codewords from the functional circuit.

1. INTRODUCTION

With the increasing complexity of VLSI circuits most of the errors have been found to be transient in nature. It is possible to detect such errors concurrently with normal operation with the aid of self-checking circuits. Of course, the errors due to permanent faults are also detected by these circuits. This has resulted in the need to obtain efficient designs for self-checking circuits.

A self-checking circuit consists of a functional circuit and a checker. The functional circuit has encoded inputs and outputs. The checker monitors the outputs of the functional circuits, and concurrently indicates an error whenever a non-codeword is detected.

The concept of totally self-checking (TSC) circuits was introduced in [1] and generalized in [2]. This concept was later extended to the strongly fault-secure (SFS) and strongly code-disjoint (SCD) concepts [3,4]. SFS and SCD circuits have the ability to function properly even in the presence of undetected faults. However, the SCD concept can be used for checkers only if its outputs are directly observable. We have generalized this concept recently to a strongly self-checking (SSC) checker [5,6]. This concept is valid even when the checker outputs are not directly observable. This may be the case when there is more than one checker in the system, and the checker outputs are reduced to two final outputs by feeding them to a two-rail checker.

It has been found that in VLSI circuits the most likely error is of the unidirectional type [7]. Thus, the codes used for encoding the inputs and outputs of self-checking circuits are usually those which can detect unidirectional errors. Various systematic and separable codes are known which can detect such errors [8]-[15].

The problem with the TSC checker designs is that they can be guaranteed to be TSC only if they get all the needed codeword tests from the functional circuit. Frequently, this requirement cannot be met.

In this paper we will give a general design method for obtaining sufficiently strongly self-checking (SSSC) checkers for any systematic or separable code which detect unidirectional errors. These checkers provide a very high level of protection against faults even though they do not receive all the needed codeword tests from the functional circuit.

2. DEFINITIONS

A. Systematic and Separable Codes

We will first formally make a distinction between systematic and separable codes. Suppose that the information symbol has k bits and the appended check symbol has r bits. Thus, a codeword has n = k + r bits.

Definition 1: A set of binary n-tuples C is called a systematic code if (a) it contains 2^n n-tuples, 0 < k < n, and (b) C = {x | x = IS, CS, where IS is the information symbol containing k bits and CS is the check symbol containing r bits, and each one of the 2^n information symbols appears as a part of some codeword). For a separable code if (a) it contains s < 2^n n-tuples, 0 < k < n, and (b) C = {x | x = IS, CS, where IS and CS have k and r bits respectively, and each one of the s information symbols appears as a part of some codeword).

Definition 3: A systematic (separable) code is called a complete systematic (separable) code if and only if each one of the 2^n r-tuples is used as a check symbol for some codeword. Otherwise, it is called an incomplete systematic (separable) code.

B. Self-Checking Circuits

A TSC functional circuit has the self-testing and fault-secure properties, whereas a TSC checker has the code-disjoint property in addition [1],[2]. The self-testing property ensures the detection of all modeled faults by codewords, while the fault-secure property implies that an incorrect codeword is never produced in the presence of a fault. The code-disjoint property implies that codewords (non-codewords) at the inputs are mapped to codewords (non-codewords) at the outputs.

It was mentioned in [16] and then in [4] that the fault-secure property is not really required in a checker. The reason given was that it does not matter if an incorrect code output is produced by the checker in the presence of a fault. This reasoning, however, is valid only if the two outputs of the checker are directly observable. As has been pointed out in [17], a system usually consists of many checkers whose outputs are checked by a final two-rail checker. For the final checker the fault-secure property is not needed. However, for the other checkers the fault-secure property is still required, otherwise the final checker may not get all its tests and, hence, may not be self-testing. To satisfy the above requirement we combine the...
testing and fault-secure, or (b) the circuit is fault-secure and always maps non-codewords at the inputs to non-codewords at the outputs, and if another fault from F occurs then either property (a) or (b) is true for the fault sequence.

Definition 6: A checker is said to be embedded if the set of codewords it receives from the functional circuit under normal operation is not enough to make it self-testing.

A checker seldom receives all the possible codewords from the functional circuit. Sometimes all the modeled faults in the checker can be detected even by a subset of the code space. However, frequently, all the faults cannot be detected by the available code-words. In some cases it is possible to redesign the embedded checker to handle only the faults that are available. However, in general, it is not possible to do so. In this situation if we still want to obtain adequate protection from the embedded checker, we need to design it according to the following definition:

Definition 7: G is sufficiently strongly self-checking (SSSC) with respect to F if before the occurrence of any fault, G is code-disjoint, and for every fault in F, either (a) the circuit is self-testing and fault-secure, or (b) the circuit is fault-secure and always maps non-codewords at the inputs to non-codewords at the outputs.

We can see that the SSSC property is a weaker version of the SSC property.

3. EMBEDDED CHECKERS FOR SYSTEMATIC CODES

In this section we will give designs for SSSC embedded checkers for systematic codes which detect unidirectional errors. Consider the checker in Fig. 1. This is a well-known structure used for the design of checkers for systematic codes, and is referred to as the normal checker [18,19].

Suppose that all the $2^k$ possible codewords are fed to this checker from the functional circuit. Then as shown in [4] the normal checker is SCD irrespective of the design of the checkbits complement generator. Of course, it is assumed that the two-rail checker gets all its tests - a condition which is very easy to satisfy. But for an SCD checker, the fault-secure property is not ensured. However, it is easy to show that the normal checker is fault-secure as well. Therefore, the normal checker can also be shown to be SCD. However, this result is not valid when the checker is embedded. We consider embedded checkers next.

A. Structure of the SSSC Embedded Checker

The general structure of our SSSC embedded checker is shown in Fig. 2. Let IS be some k-bit information symbol and CS the corresponding r-bit check symbol. When IS is the input vector to the PLA then the output vector is the two-rail codeword $(CS,CS)$ (CS is the bitwise complement of CS). The correctness of the two-rail code at the PLA outputs is verified using an $r$-variable two-rail checker. This is done on the first phase $\phi_1$ of the clock. On the second phase $\phi_2$ of the clock the $r$-variable two-rail checker is used again to verify if the r-checkbits from the functional circuit and one of the r-bit rails from the PLA form a two-rail code. Henceforth, the checker in Fig. 2 will be referred to as the general checker. A similar structure was also used in [22], although in a different context. The fault model for this checker consists of stuck-at, bridging and crosspoint faults in the PLA and stuck-at faults in the two-rail checker. It was shown in [20] that the single faults in the PLA can only cause unidirectional errors. The stuck-at faults at the external inputs of the PLA are assumed to belong to the functional circuit, not to the checker, as these faults can cause non-unidirectional errors [20]. However, this assumption does not lead to any loss of generality as we will see later. The clock lines $\phi_1$ and $\phi_2$ are assumed to be fault-free.

B. Making the Two-Rail Checker Self-Testing

It is well-known that an $r$-variable two-rail checker can be designed using a tree of two-variable two-rail checkers. If designed in this fashion the $r$-variable two-rail checker can be made self-testing with the help of only four codewords. Conversely, if the set of vectors that will be fed to the two-rail checker is known, one can generally design the checker to be self-testing, and, hence, TSC [21].

The codes in [12]-[14] are complete systematic codes. Thus, a TSC design of the $r$-variable two-rail checker for these codes is straightforward. There are some rare incomplete systematic codes for which the above method may not be successful. For example, the two-rail checker cannot be made self-testing when the outputs of the functional circuit are encoded with a Berger code for which $k = 2^r - 1$. However, such a code can be converted into an equivalent complete systematic code [18]. Similarly, if the above problem arises for a separable code, we can use the method in [22] to arrive at a complete separable code.

C. Use of Systematic Codes for Embedded Checkers

Consider the Berger encoding given in Table 1. For this code, $k = 3$. Hence, $r = \lceil \log(2^{k+1}) \rceil = 2$. Note that this value of $r$ is derived on the basis of the assumption that all the possible information symbols are present. However, when this assumption is not true, as in Table 1, then a systematic code can still be used, although it may sometimes be more efficient to use a separable code. We shall give an example later in Section 4 to illustrate this.

Table 1. Berger code for a set of information symbols

<table>
<thead>
<tr>
<th>Information Symbol</th>
<th>Zero-count Check Symbol</th>
</tr>
</thead>
<tbody>
<tr>
<td>000</td>
<td>3</td>
</tr>
<tr>
<td>001</td>
<td>2</td>
</tr>
<tr>
<td>010</td>
<td>1</td>
</tr>
<tr>
<td>110</td>
<td>0</td>
</tr>
</tbody>
</table>

If the functional circuit produces only the five codewords derived from the above table, then it may not be possible to make the checker TSC or SSC.

It is easy to see that one can obtain an equivalent complete systematic code to the Berger code as long as there are at least 2 $r$ information symbols to be encoded, should it be necessary to do so. This way the two-rail checker can always be assured of getting all the required codewords. The same is, of course, also true for other systematic and separable codes as well.

D. Encoding of the PLA Outputs

We map the information symbols, which the PLA receives during normal operation, to two-rail codewords at the outputs. The information symbols, which are not normally received during fault-free operation, are mapped to $r$-out-of-$2^r$ codewords (which are not also two-rail codewords) at the outputs. It is easy to see that the $r$-variable two-rail code space is always a subset of the $r$-out-of-$2^r$ code space.

Let us look at the information symbols in Table 1 again. If these are the only ones received normally by the PLA then the PLA outputs can be encoded as shown in Table 2.

Table 2. PLA encoding

<table>
<thead>
<tr>
<th>Information Symbol</th>
<th>First Rail</th>
<th>Second Rail</th>
</tr>
</thead>
<tbody>
<tr>
<td>000</td>
<td>11</td>
<td>50</td>
</tr>
<tr>
<td>001</td>
<td>10</td>
<td>01</td>
</tr>
<tr>
<td>010</td>
<td>10</td>
<td>01</td>
</tr>
<tr>
<td>011</td>
<td>01</td>
<td>01</td>
</tr>
<tr>
<td>100</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>101</td>
<td>01</td>
<td>01</td>
</tr>
<tr>
<td>110</td>
<td>01</td>
<td>10</td>
</tr>
<tr>
<td>111</td>
<td>00</td>
<td>11</td>
</tr>
</tbody>
</table>
After the assignment is done, an SFS PLA can be realized by using the method in [20]. It is clear that if the input pattern to the embedded general checker is not one of the normally received codewords, then an error indication would be given at \((z_1,z_2)\) on \(\phi_1\). The above stems from the following remark.

Remark 1: For embedded checkers it is valid to define the code-disjoint property with respect to the normally occurring input code space, and not the whole input code space.

Theorem 1: The general embedded checker designed for the systematic code is SSSC.

Proof: We have to first show that the checker is code-disjoint when no fault is present. If \((IS,CS)\) is a normally received codeword from the functional circuit, then the \(r\)-variable two-rail checker will receive a two-rail codeword on both \(\phi_1\) and \(\phi_2\). Thus, \((z_1,z_2)\) will also have a 1-out-of-2 codeword. Now suppose \((IS,CS)\) is not one of the codewords normally received by the embedded checker. There can be an error in \(IS\) or \(CS\) or both. If the error is in \(IS\) and is such that it has turned into an information symbol not normally received by the PLA, then the corresponding PLA output would be an \(r\)-out-of-2 codeword, which is also a two-rail codeword. Since the two-rail checker is code-disjoint, this would result in a non-codeword at \((z_1,z_2)\) on \(\phi_1\). If the error in \(IS\) is such that it has turned into another information symbol normally received by the PLA, then \(IS\) and \(CS\) will not correspond to \(\phi_1\). Thus, the vector fed to the two-rail checker will not be a two-rail codeword on \(\phi_1\). This will be detected at \((z_1,z_2)\). Similarly, if in addition, \(CS\) was also in error or if \(CS\) alone was in error, then again the vector fed to the two-rail checker will not be a two-rail codeword, at least on \(\phi_2\). This stems from the fact that an error cannot change one normally received codeword into another normally received codeword. Thus, the checker is code-disjoint in the fault-free case.

We will now prove that the checker is either fault-secure and self-testing in the presence of a fault, or, if the fault is not detected by available input codewords, then the checker is fault-secure and code-disjoint in spite of the fault. The following cases arise:

Case 1: \(f_{\text{IS}}\) present

Since the two-rail checker is TSC it is by definition fault-secure and self-testing with respect to any fault \(f_{\text{IS}}\) in it. Therefore, the embedded checker would also be fault-secure and self-testing with respect to \(f_{\text{IS}}\).

Case 2: \(f_{\text{CS}}\) present

If the fault in the PLA, denoted \(f_{\text{CS}}\), is exercised by a normally received input codeword \((IS,CS)\) then it will be detected at \((z_1,z_2)\) on \(\phi_1\). Also, due to the fault-secure nature of the PLA, the input codewords which do not detect \(f_{\text{CS}}\) would map to the correct codeword at \((z_1,z_2)\). This would be true even if \(f_{\text{IS}}\) were undetectable by any available input codewords. Now suppose that \((IS,CS)\) is not a normally received codeword or else is a non-codeword. If \(IS\) exercises \(f_{\text{CS}}\) then the output of the PLA will not be a two-rail codeword. If \(IS\) is not a normally received information symbol and does not exercise \(f_{\text{CS}}\) then the PLA output will be an \(r\)-out-of-2 codeword. In either case, \((z_1,z_2)\) will have a non-codeword on \(\phi_1\). If \((IS,CS)\) is any other kind of non-codeword then there will be a non-codeword at \((z_1,z_2)\) on \(\phi_2\).

From the above arguments the embedded checker is SSSC.

E. Protection Provided by the SSSC Checker

We saw earlier that an embedded checker, by definition, does not meet the TSC goal. Thus, our aim is to maximize the protection provided by such checkers as much as possible. We next show that the protection provided by the general checker is even greater than that suggested by Theorem 1.

Let \(f_{\text{CS}}\) denote multiple faults in the PLA and \(f_{\text{IS}}\) denote a fault in the functional circuit. Consider the following cases:

1. Fault sequence \(f_{\text{IS}}\) present

Although not explicitly stated in [20], if the PLA does not receive all the required information symbols it cannot be guaranteed to be SFS if an undetected fault \(f_{\text{CS}}\) is followed by another fault \(f_{\text{IS}}\). This is owing to the fact that \(f_{\text{CS}}\) may change the PLA function. But this fact may not be detected due to the unavailability of all the required information symbols. In this case the arguments given in [20] are no longer valid. However, for our checker this does not create any problems, as explained below.

Suppose the fault sequence \(f_{\text{IS}} < f_{\text{CS}} < f_{\text{IS}}\) occurs, such that \(f_{\text{CS}}\) is undetectable by any normally received information symbol.

After \(f_{\text{IS}}\) occurs, if \(f_{\text{CS}}\) becomes detectable, then it will be detected at \((z_1,z_2)\) on \(\phi_1\). If the PLA becomes non-SFS then a normally received codeword could result in an incorrect two-rail codeword at the PLA output. Although not detectable by \(f_{\text{CS}}\), this will be detectable by \(f_{\text{IS}}\). Another possibility is that a normally received codeword may result in an incorrect \(r\)-out-of-2r word which is not also a two-rail codeword at the PLA output. This will be detected on \(\phi_1\). Thus, it does not matter if \(f_{\text{CS}}\) gets detected first or the PLA becomes non-SFS first. Now suppose that \(f_{\text{IS}}\) is undetectable by any available codeword, but the PLA remains fault-secure in its presence. But in this case if the input pattern \((IS,CS)\) is a normally received codeword, then \((z_1,z_2)\) will have the correct codeword. The above arguments can be applied even if \(f_{\text{CS}}\) has more than two faults. Let \(f_{\text{CS}} = < f_{\text{CS}1}, f_{\text{CS}2}, \ldots, f_{\text{CS}m} >\). When the last fault \(f_{\text{CS}m}\) occurs, either \(f_{\text{CS}m}\) becomes detectable, or we can deduce that the PLA, and hence the general checker, remains fault-secure with respect to \(f_{\text{CS}}\).

(2) Fault sequence \(f_{\text{IS}} < f_{\text{CS}1}, f_{\text{CS}2}, \ldots, f_{\text{CS}m} >\) present.

The faults present in the PLA before \(f_{\text{IS}}\) occur must be undetectable. As shown above, their presence does not adversely affect the operation of the PLA. Since the PLA is fault-secure it continues to supply the two-rail checker with all the required codewords. Thus, when the latest fault \(f_{\text{IS}}\) occurs it is detected at \((z_1,z_2)\) on both \(\phi_1\) and \(\phi_2\).

(3) Fault sequence \(f_{\text{IS}}, f_{\text{CS}}\) present.

The faults present in the PLA before \(f_{\text{CS}}\) occur must be undetectable, and the PLA must be fault-secure from previous arguments. If \(f_{\text{IS}}\) is also undetectable then the situation is the same as in Case 1. Note that this situation can arise if the functional circuit is SFS and is redundant with respect to \(f_{\text{IS}}\). Hence, assume that \(f_{\text{IS}}\) is detectable and produces a non-codeword \((IS,CS)\) at the output of the functional circuit. If \(IS\) is a normally received information symbol then the multiple fault will be detected on \(\phi_1\). Now suppose that \(IS\) is not a normally received information symbol. If \(IS\) exercises \(f_{\text{CS}}\) then this multiple fault will be detected on \(\phi_1\). Another possibility is that \(IS\) produces an incorrect codeword at the PLA output. Then it is possible that a codeword will be produced at \((z_1,z_2)\) on both \(\phi_1\) and \(\phi_2\), thus destroying the code-disjoint property.

From the above arguments we can conclude that our SSSC embedded checker provides a very high level of protection.

4. EMBEDDED CHECKERS FOR SEPARABLE CODES

In the case of separable codes, all the \(2^n\) information symbols do not occur. However, the output space is divided into only two categories: code or non-code. The redundancy of the separable codes is always less than or equal to that of systematic codes [9],[13], for the same unidirectional error-detecting capability.

Consider the 4-bit information symbols shown in Table 3. Suppose we want an all-unidirectional error-detecting code. If we use the Berger code [8] we will require 3 checkbits, but if we use the Smith code [9] we need only 2 checkbits.

<table>
<thead>
<tr>
<th>Information Symbol</th>
<th>Berger Check Symbol</th>
<th>Smith Check Symbol</th>
</tr>
</thead>
<tbody>
<tr>
<td>0000</td>
<td>001</td>
<td>11</td>
</tr>
<tr>
<td>0001</td>
<td>011</td>
<td>10</td>
</tr>
<tr>
<td>0100</td>
<td>011</td>
<td>10</td>
</tr>
<tr>
<td>1001</td>
<td>010</td>
<td>01</td>
</tr>
<tr>
<td>0111</td>
<td>001</td>
<td>00</td>
</tr>
<tr>
<td>1101</td>
<td>001</td>
<td>00</td>
</tr>
</tbody>
</table>
Encoding of the PLA Outputs

Even when the output of the functional circuit is encoded using a separable code, we can use the method in Section 5 for the PLA encoding. Corresponding to the information symbols which are used during normal operation we have two-rail codewords, and corresponding to the other information symbols we have r-out-of-2r codewords, which are not two-rail codewords.

Theorem 2: The general embedded checker designed for the separable code is SSSC.

The proof of this theorem is similar to the proof of Theorem 1. The discussion in Section 3.E is valid for separable codes as well.

5. CONCLUSIONS

Many self-checking checker designs have been presented in literature for various unidirectional error-detecting codes. However, these designs are guaranteed to be self-checking only if certain assumptions are made. One of these assumptions is that the checker should receive a set of codeword tests which can flush out all the detectable faults. However, for real-life VLSI circuits, frequently, the functional circuit does not provide the checker with enough codewords for detecting all the detectable faults. For such embedded checkers the TSC goal can not be guaranteed to be met. Thus, the best we can do is to maximize the protection provided by the checker. This was precisely our aim in defining and designing the SSSC embedded checker in this paper.

REFERENCES