A Methodology for the Design of SFS/SCD Circuits for a Class of Unordered Codes

SANDEEP PAGEY
Department of Computer Science and Engineering, Indian Institute of Technology, Bombay 400 076, India

S.D. SHERLEKAR
Department of Computer Science and Engineering, Indian Institute of Technology, Bombay 400 076, India

G. VENKATESH
Department of Computer Science and Engineering, Indian Institute of Technology, Bombay 400 076, India

Editor: C. Kime

Abstract. We present two design methods that produce concurrently testable and cascadable combinational blocks for a given logic function. In the first method, the designed block is strongly fault-secure and code-disjoint. Any unordered coding scheme can be used for the input and output. The second method produces designs that are strongly fault-secure and strongly code-disjoint. Here the encoding requires some simple density properties that are seen to be satisfied by the commonly used coding schemes. This makes the method applicable to a larger class of coding schemes than the existing methods. We also show that our designs have lower hardware overhead.

Key words: Self-checking circuits, unordered codes, strongly fault-secure, strongly code disjoint, concurrent error detection.

1. Introduction

The design of self-checking circuits realizing a given logic function has attracted the attention of researchers for over two decades. The design of a self-checking circuit requires that its input and output be coded using a suitable error detecting code [1]. Anderson's work [2] provided a consolidation and formalization of the properties required to be satisfied by circuits which are self-checking. In particular, two properties, self-testing and fault-secure, were identified as the two characteristics of a totally self-checking circuit. While these properties are applicable for designing circuits realizing any logic function, most of the available literature deals with the design of self-checking checkers. A checker is a circuit realizing a function that checks whether, for a given coding scheme, its input is a valid codeword [3], [4], [5], [6], [7], [8], [9], [10]. Such circuits are required to be totally self-checking so that a fault in the circuit can also be detected. In addition, such circuits are required to be code disjoint, i.e., they should produce codeword outputs for codeword inputs and non-codeword outputs for non-codeword inputs.

Smith and Metze [11] showed how the “TSC goal” can be achieved with a wider class of circuits called strongly fault-secure (SFS) circuits, which may not be self-testing. These circuits need only have the property that for any sequence of faults in the circuit, the first erroneous output is a noncodeword, provided that between the occurrence of two faults in a sequence, the circuit is sufficiently exercised, i.e., it receives all the necessary input codewords for testing a fault before the next fault occurs.

Smith and Metze [11] also gave a methodology for the design of SFS circuits to realize any given logic function, provided the inputs and outputs are coded using an unordered coding scheme [3], [6], [12], [13], [14]. The methodology is based on the fact that any function mapping one unordered code space to another can be realized using positive monotone functions [15], which satisfy the SFS property. The fault set assumed by Smith and Metze is the set of all multiple unidirectional stuck-at faults. Subsequently, attempts were made to give methodologies for the design of strongly fault secure and code disjoint (CD) circuits [16].
Nicolaidis [17] introduced the notion of strongly code disjoint (SCD) circuits which generalize the CD property just as SFS generalizes the TSC property.

Nanya [18] observed that circuits must enjoy the SCD property (in addition to the SFS property) so as to be cascadable to build larger circuits. Nanya [18] also gave a methodology for designing SFS/SCD circuits for a restricted class of unordered codes which have to be separable, systematic, and complete. This, in particular, leaves out coding schemes such as m-out-of-n codes.

In this article, we present a methodology for the design of SFS/CD and cascadable SFS/SCD circuits to realize any given logic function. The fault set we consider consists of all multiple unidirectional stuck-at faults. For SFS/CD, we only require that the input and output code spaces be unordered. For SFS/SCD we also require the code spaces to possess two additional simple properties which we call density properties. Many interesting coding schemes such as maximal Berger code, m-out-of-n code and two rail code possess these properties and as such our requirements are less restrictive than those of Nanya [18]. Our methodology also has less overheads in terms of hardware. These SFS/SCD circuits can be cascaded to form bigger SFS/SCD circuits provided each interface is sufficiently exercised.

This article is organized as follows. In Section 2, we introduce all the definitions and notations used in the article. Section 3 outlines the methodology used by Smith and Metze [11] to realize SFS circuits for unordered codes. We also introduce the property of weakly code disjoint (WCD) circuits and show that the circuits realized using the methodology of Smith and Metze are also WCD. In this section we also show how to design SFS/WCD circuits in such a way that they can easily be modified to obtain SFS/CD circuits. The required modification is explained in Section 4. This can be carried out for any unordered input and output code spaces. Section 5 introduces a class of unordered codes and gives a general methodology for designing SFS/SCD circuits for this class. Section 6 determines the hardware overheads of our method to realize cascadable SFS/SCD circuits and compares them with those of earlier methods. Section 7 concludes the article. Some of the proofs are relegated to an appendix.

2. Definitions and Notations

This section defines the various terms and notations used in this article.

**Definition 1.** A circuit is self-testing (ST) [2] for a fault set \( F \) iff, for every fault in \( F \), the circuit produces a non-codeword output for at least one codeword input.

**Definition 2.** A circuit is fault-secure (FS) [2] for a fault set \( F \) iff, for every fault in \( F \), the circuit never produces an incorrect codeword output for any codeword input.

**Definition 3.** A circuit is totally self-checking (TSC) [2] for a fault set \( F \) iff the circuit is both ST and FS for \( F \).

**Definition 4.** A circuit is code disjoint (CD) [2] iff it always maps codeword inputs to codeword outputs and non-codeword inputs to non-codeword outputs.

**Definition 5.** A circuit is strongly fault-secure (SFS) [11] for a fault set \( F \) iff, for every fault \( f \) in \( F \), either
- the circuit is TSC for \( \{f\} \), or
- the circuit is FS for \( \{f\} \) and if fault \( f \) occurs, the resultant circuit is still SFS for \( F - \{f\} \).

**Definition 6.** A circuit is strongly code disjoint (SCD) [17] for a fault set \( F \) iff, for every fault \( f \) in \( F \), either
- the circuit is CD, and is ST for \( \{f\} \), or
- the circuit is CD, and if fault \( f \) occurs, the resultant circuit is still SCD for \( F - \{f\} \).

In the following text, for any two vectors \( x = (x_1, \ldots, x_n) \) and \( y = (y_1, \ldots, y_n) \), \( N(x, y) \) represents the number of 1 to 0 crossovers from \( x \) to \( y \), i.e., \( N(x, y) \) is the cardinality of the set \( \{i \mid x_i = 1 \text{ and } y_i = 0\} \).

**Definition 7.** A vector \( a \) is said to be covered by a vector \( b \), denoted by \( a < b \), if \( N(a, b) = 0 \) and \( N(b, a) \geq 1 \). The vector \( b \) is said to be covering the vector \( a \).

**Definition 8.** A vector \( w \) is said to be immediately covered by a vector \( z \), denoted by \( w <^* z \), if \( N(w, z) = 0 \) and \( N(z, w) = 1 \). The vector \( z \) is said to be immediately covering the vector \( w \).

**Definition 9.** Two vectors \( x \) and \( y \), \( x \neq y \), are said to be unordered if \( x \not< y \) and \( y \not< x \).

**Definition 10.** A code space \( C \) is defined to be unordered if

\[ \forall a, b \in C \land a \neq b; a \not< b \land b \not< a. \]