FACTA UNIVERSITATIS (NIŠ)
SER.: ELEC. ENERG. vol. 17, April 2004, 69-79

On VHDL Synthesis of Self-Checking Two-Level Combinational Circuits

Tatjana R. Stanković, Mile K. Stojčev, and Goran Lj. Djordjević

Abstract: Concurrent error detection (CED) is an important technique in the design of system in which dependability and data integrity are important. Using the separable code for CED has the advantage that no decoding is needed to get the normal output bits. In this paper, we address the problem of synthesizing totally self-checking two-level combinational circuits starting from a VHDL description. Three schemes for CED are proposed. The first scheme uses duplication of a combinational logic with the addition of a totally self-checking comparator. The second scheme for synthesizing combinational circuits with CED uses Bose-Lin code. The third scheme is based on parity codes on the outputs of a combinational circuit. The area overheads and operating speed decreases for seven combinational circuits of standard architecture are reported in this paper.

Keywords: Concurrent error detection, totally self-checking circuits, error-detecting codes, VHDL.

1 Introduction

A digital system consists of logic circuits that can fail during normal operation. After a system fails, engineers must identify any faulty circuits and replace them. An ideal solution to dealing with faulty systems is to be able to rapidly identify a fault and have a backup system to take over. To implement this solution at macro-level, we would use combinational circuits that are self-checking.

Self-checking circuits (*SCC*s) are intended to: 1) detect the presence of intermittent, transient and permanent faults; and 2) signal the presence of errors at the output of the circuit immediately after their occurrence, thus preventing their propagation throughout the system [1]. The *SCC*, see Fig. 1, consists of a functional

Manuscript received 23 December, 2003.

The authors are with Faculty of Electronic Engineering, University of Niš, Beogradska 14, 18000 Niš, Serbia and Montenegro (e-mail: [tatjanas,stojcev,gdjordj]@elfak.ni.ac.yu).

circuits, F, which produces encoded output vectors, and a checker, C, which checks the vector to determine if an error has occurred. In particular, F should be designed in such a way that, in the fault free case, its output belongs to a chosen error detecting code, while C should verify whether or not this is the case. The functional circuit can be either combinational or sequential. A self-checking system consists of an interconnection of *SCC*s [2, 3, 4].

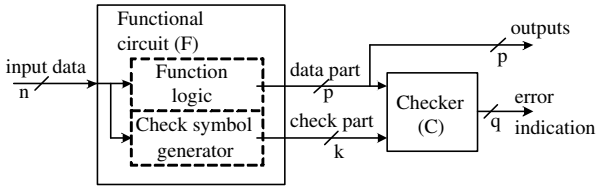

Fig. 1. General structure of self-checking circuit for separable code

The concept of self-checking is formalized by several definitions (see Section 2.) out of which those of totally self-checking (*TSC*) [1] and strongly fault-secure (*SFS*) circuits [5] have played a basic role. In fact, it has been demonstrated that, as far as single stuck-at faults are concerned, for a *SCC* to behave correctly (i.e., either to produce the correct functional output or to give an error message), C should be *TSC*, while F should be *SFS* with respect to internal faults. The stuck-at fault error as the most common type of unidirectional error in VLSI ICs [4] is said to occur when there are multiple transitions in either 0r1 or the 1r0 direction, but not both. As a consequence, different works have been dedicated to design rules and coding techniques making combinational [2, 6, 7] and sequential [2, 7] functional circuits *SFS*, and the checkers *TSC* [1, 7, 8] with respect to stack-at faults.

In the present paper, we have considered implementation of three different coding techniques (duplication of function, parity check codes, and Bose-Lin codes), for the purpose of concurrent error detection (*CED*), into seven combinational circuits with standard architecture in order to make these circuits self-checking. In addition, the practicality of the design in term of area overhead, maximal operating frequency, and suitability to implement the self-checking system with FPGA and CPLD VLSI ICs using VHDL - Active HDL V. 3.5 and Xilinx Project Navigator V. 4.2WP2.x as tools for designing, synthesizing and simulating these circuits is discussed.

2 Background

Concurrent (on-line) error detection techniques used in digital systems can be divided into two classes: circuit-level and system-level techniques. The use of single

error correcting and double error detecting codes for memories, parity bits for data buses, residue (codes), Berger (codes), Bose-Line (codes) and *m*-out-of-*n* codes for arithmetic, and self-checking sequential-circuits, are all examples of circuit-level techniques. Capability-based addressing, watchdog timers, fault-tolerant data structure and use of replication, such as *FTMP* and *SIFT*, and *N*-version programming are some of the examples of the techniques used to detect errors at the system level [2, 9, 10].

Usually, incorporation of techniques for concurrent error detection at the system level lead to considerable hardware overhead, performance degradation, or error detection latency. In particular, their impact on circuit speed may not be tolerable for high performance applications [10]. In designs using circuit-level technique the correctness and complexity of the overall operation critically depends on the correctness of the control logic and the self-checking circuit. A lot of work has been done in the area of self-checking checker design for different codes [1, 2, 3, 6, 11].

The process of appending check bits to the information bits is called encoding, while the opposite process-extracting the original information bits from a code-word- is known as decoding. In general, any set of objects can be represented by a set of bit strings within which the different bit strings represent the different objects. The set of bit string is called a code, and a particular bit string is called a code-word. Any given *n*-bit code can be regarded as a subset of all possible *n*-bit strings. Strings, included in that particular subset are called code-words while strings not included are called noncode-words. A code is called an error-detecting code if it has the property that certain types of errors that affect bits of a code-word will change the code-word into a noncode-word.

In addition, code can be classified as either separable or nonseparable. A separable code (also called systematic code) is a code in which code-words are constructed by appending check bits to the normal output bits. Three types of systematic codes that are used for concurrent error detection are Bose-Lin codes, Berger codes and parity-check codes. The separable nature of these codes facilitates the derivation of efficient self-checking checkers. In general, the Berger coding design technique for the purpose of the concurrent error detection utilizes a set of full-adder modules providing summation of the information bits to produce the binary representation of the number of ones (zeros) in the information. Bose-Lin codes are an efficient solution for providing detection of up to *t* unidirectional errors. They are systematic codes and require a fixed number of check bits, independent of the number of information bits. These two properties make Bose-Lin codes an efficient solution for synthesizing arbitrary circuits with *CED* capability. The codes are constructed by counting the number of ones or zeroes, similar to Berger codes. The count is then modified depending on code parameter *t*. For `t=2` and `t=3`, the counts are performed by *modulo* 4 and 8, resulting in 2 and 3 check bits, respec-

tively [11]. Bose-Lin codes with check-bits greater than 3 are explained in details in [1]. Bose-Lin checkers have a simple structure as they are based on modulo operations [8, 11]. Parity check-code is a code with the single parity bit equal to the check part of the code-word. Parity checker uses a simple structure that is realized as a combinational digital network which computes the *sum modulo* 2 of the code-word bits.

3 Definition of Self-Checking Circuits

*SCC*s are increasingly becoming a suitable approach to the design of complex VLSI ICs, to cope with the growing difficulty of on-line and off-line testing. *SCC*s are class of circuits in which occurrence of fault can be determined by observation of the outputs of the circuits. An important subclass of these self-checking circuit is known as "totally self-checking" (*TSC*) circuits. The fault model assumed for this paper recalls some basic definitions on the design of *TSC* circuits and the properties that need to be guaranteed [1]:

Definition 1 *A combinational circuit is self-testing for a fault set F if and only if (*`iff`*), for every fault in F, the circuit produces a noncode-word output during normal operation for at least one input code-word. Namely, if during normal circuit operation any fault occurs this property guaranties an error indication.*

Definition 2 *A combinational circuit is strongly-fault-secure (SFS) for a fault set F* `iff`*, for every fault in F, the circuit never produces an incorrect output code-word.*

Definition 3 *A combinational circuit is SCC for a fault set F* `iff`*, for every fault in F, the circuit is both self-testing and strongly-fault-secure.*

Definition 4 *A circuit is code-disjoint for a fault set F* `iff` *input code-words map onto output code-words and input noncode-words map onto output noncode-words. Self-checking can be defined as the ability to verify on-line whether there is any faults in control logic (within the VLSI IC) without the need for externally applied test stimuli.*

Definition 5 *A TSC circuit is a TSC checker for a fault set F* `iff`*, for every fault in F, the circuit is self-testing, strongly-fault-secure, and code-disjoint. Namely, the checker is said to be TSC circuit* `iff`*, under the fault model assumed: a) The output of checker is 01, or 10 whenever the input is a code-word (strongly-fault-secure property); b) The output is 00 or 11 whenever the input is not a code-word (code-disjoint property); and c) Fault in the checker are detectable by test inputs that are code-words and under fault-free condition, for at least two inputs X and Y, the*

checker outputs are 01 and 10, respectively (self-testing property). Furthermore, the output of the checker is 00 or 11 whenever a fault is said to be detected by an input.

4 Design Strategies

Hardware-, information- and time-redundancy are three widespread strategies for VLSI ICs with concurrent error detection properties [10].

Information-redundancy, as a first approach, involves the use of coding techniques that enhance circuit capability for reliable operation. Memory-, data communication-, arithmetic-, and control-modules can be made reliable through the use of error detecting and correcting codes. Numerous codes such as Berger codes, Bose-Lin codes, residue codes, parity check codes, cyclic codes and others, have been proposed for storage-, data transfer-, data manipulating-, and data control-functions [10]. In some cases where performance is not a bottleneck, a second approach called time-redundancy can be used. It involves the use of the same hardware repeatedly in time for the same inputs and comparing the results. Examples of such approaches are recomputing with shifted operands, alternating logic, etc. [10]. In fact these are restricted approaches applicable to only certain specific operations. A third approach to increases the reliability of circuits is to use hardware redundancy. The simplest hardware redundancy approach to designing a *TSC* circuit is duplication. Typically, the design implements two copies of the same circuit. The second copy produces output values complementing the value of the first copy, and a tree of two-rail code (*TRC*) checkers makes a bitwise comparison of the outputs. Whenever the natural and complementary outputs configurations differ from each other, or whenever a fault affects one of the self-checking *TRC* checkers, the error signal reports the presence of fault. The advantage of this approach is that is applicable to any general function. Unfortunately, with duplication and comparison the area overheads are too high (more than 100%).

The simplest scheme for error detection is parity checking. Synthesis techniques for generating multilevel circuits with concurrent error detection based on parity check codes were presented in [2, 9, 12, 13]. Efficient schemes have been developed for concurrent error detection in circuits with regular structures, as for example adders [14], and multipliers [9].

In [11] a procedure for synthesizing multilevel circuits with concurrent error detection based on Bose-Lin codes was considered. There are two structural constraints on the circuit that, if satisfied will guarantee detection of all internal faults with a Bose-Lin code. The first constraint is that the circuit must be inverter-free so that only unidirectional errors can occur. The second constraint is that no non-input

node in the circuit can have a path to more than t outputs if a t unidirectional error detecting Bose-Lin code is used. Synthesis and layout of combinational circuits with *CED* based on Bose-Lin codes show low hardware overheads, making it a practical solution [11].

In this paper we use a methodology to design of totally self-checking systems specified in VHDL. At the start the proposed method requires from a user to specify the logical function of the original logic module in Boolean form. Then we enrich the functional specification with check symbol generator and checker parts. After that the expanded specification is converted into VHDL description and synthesized with commercial tools. For implementation FPGA or CPLD technology is used.

5 Implementation

The procedure for synthesizing totally self-checking combinational logic circuits derived from VHDL specification has been implemented by making modifications to the following seven representatives: i) BINBCD6/8/12 - 6/8/12 bits binary to BCD converter based on SN74185A; ii) COMPAR - 4 bits magnitude comparator based on SN7485; iii) DEMUX38 - 3-line to 8-line decoder/demultiplexer based on SN74AHC138; iv) DIS18SG - 18 segment solid state alphanumeric display HDSP-6300; v) MULTIPL - 4 bits multiplier.

A. Duplication and comparison: For a given combinational logic circuit we synthesize duplicate circuit (see for example Fig. 2). The original part of the

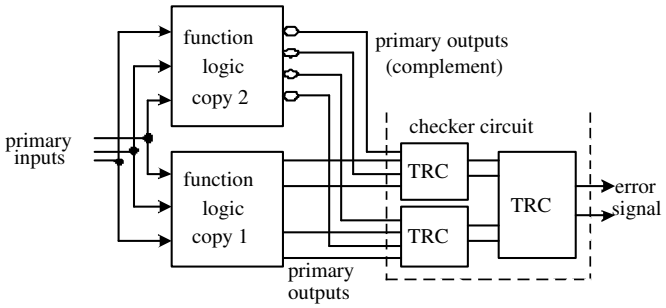

Fig. 2. Duplication of the function

circuit has true, while its copy complemented output values. Then the logic for the *TSC* comparator is synthesized. The procedure for generating logic for totally self-checking comparator is adapted to the number of signals to be compared. Finally the duplicated logic and the *TSC* equality comparator are technology mapped to VLSI ICs from Xilinx FPGA and CPLD families.

B. Using Bose-Lin code: In this scheme the Bose-Lin code as an efficient solution for providing concurrent detection of all unidirectional errors is used. The circuit (Fig. 3) with included logic for Bose-Lin code generation with order to detect any single fault inside the circuit is used. The circuit for totally self-checking equality comparator is also generated. Then the entire circuit is technology mapped to Xilinx FPGA and CPLD family chips.

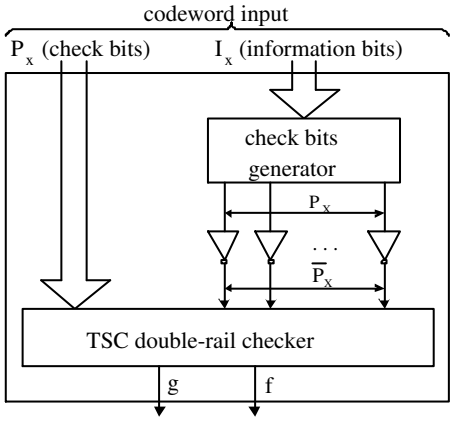

Fig. 3. TSC checker for Bose-Lin code

C. Using parity code: Given a combinational logic circuit we first partition the outputs using synthesis scheme based on the use of parity codes on the output signals of the circuit. We assume that primary inputs are fault free. Single-parity (*pov*) and multiple-parity-group synthesis techniques (*pg2* and *pg4*) are used. To make the circuit self-checking for all internal single stuck-at-faults, logic cannot be shared between two outputs in the same parity group because when a fault occurs in the shared logic, the error could propagate to both outputs causing a two bit error which would not be detected by the parity checker. The more parity groups there are, the more logic sharing is possible, however more parity groups require more parity predict logic to generate the check bits. Therefore a tradeoff exist between the number of parity groups (i.e. check bit) and the constraints on logic sharing between outputs. The circuits for parity check bit generator is synthesized, and after that the circuit for *TSC* equality comparator is synthesized too. Then the entire circuit is technology mapped to Xilinx FPGA and CPLD family chips. Block diagram of self-checking circuit that combines four pairs of *TSC* outputs into a single pair error signal is sketched in Fig. 4.

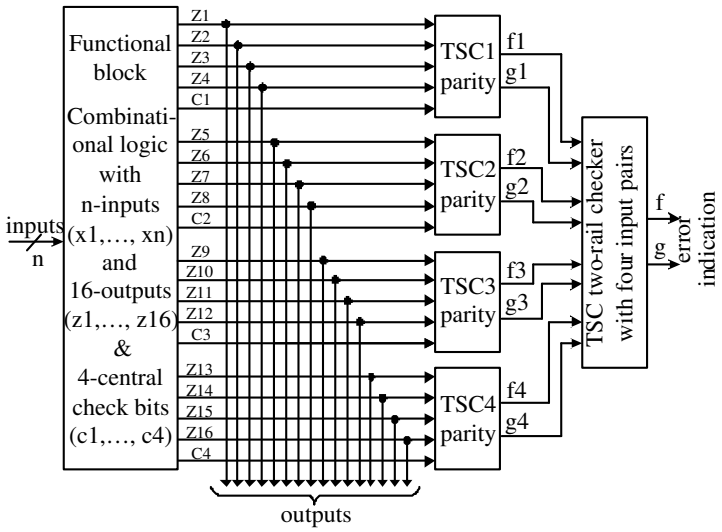

Fig. 4. Self-checking circuits for 16-bit data plus 4 parity bits

6 Results

The goal of the synthesis procedure was to select the code that will require the least area to implement self-checking principle at maximal operating frequency. The area of the circuit is equal to the sum of the area of the original function logic, *CED* logic, and checker. The area required by the original function logic depends on how much logic sharing is possible. The area required by the *CED* logic depends on the size of the checking function that must be implemented for each code. The area required by the checker depends on how many checking groups there are.

We have applied our schemes on seven different combinational circuits with standard architecture. All circuits are specified as two-level combinational logic circuits. In Table 1, the information on various circuits are presented. It relates to the case when all seven circuits of standard architecture are technology mapped to Xilinx FPGA circuit XC2S100-5 as typical representative of Spartan 2 series, and Xilinx CPLD circuit XCR3384XL-7-TQ144 as representative of CoolRunner XPLA3CPLD series. Results, presented in Table 1, that relate to the CPLD circuit are given in brackets.

We report on the results of the area overhead for all seven synthesized circuits in Table 1 with respect to the number of FPGA-slices/CPLD-macrocells, i.e. #slc/#mcl, used to implement the required scheme. We want to emphasize here the fact that, all the necessary checkers that are needed for the different schemes are generated, and the area required to implement them are included in the data

presented in Table 1. Details related to the speed of operation for all circuits of standard architecture in a form of maximum delay (pad to pad (*tPD*) delay) for both technology mapped variants are given in Table 1, also.

Table 1. Implementation of self-checking technique on FPGA XC2S100-5 device from Spartan2 and CPLD XCR3384XL-7-TQ144 device from CoolRunner XPLA3CPLD family. Notice: orig - original circuit; dup - duplication of function; Blin - Bose-Lin code; pov - single parity; pg2 - two-bit parity group; pg4 - four-bit parity group.

circuit		#slc(#mcl)	Max. Delay [ns]	Area Overhead (%)	Speed Decreasing (%)
BINBCD6	orig	6(8)	11.545(13.6)	0(0)	0(0)
	dup	19(19)	18.283(38.0)	216.7(137.5)	58.4(179.4)
	Blin	17(43)	18.322(38.0)	183.3(437.5)	58.7(179.4)
	pov	9(12)	13.335(25.3)	50(50)	15.5(86)
	pg2	18(18)	15.986(31.9)	200(125)	38.5(134.5)
	pg4	13(39)	14.521(19.7)	116.7(387.5)	25.8(44.8)
BINBCD8	orig	14(15)	15.469(31.9)	0(0)	0(0)
	dup	37(33)	25.860(74.6)	164.3(120)	67.2(133.8)
	Blin	57(54)	31.607(110.7)	307.1(260)	104.3(247)
	pov	30(33)	17.793(80.7)	114.3(120)	15(153)
	pg2	35(37)	22.589(68.5)	150(146.7)	46(114.7)
	pg4	32(36)	20.076(68.5)	128.6(140)	29.8(114.7)
BINBCD12	orig	40(48)	22.327(68.5)	0(0)	0(0)
	dup	96(90)	38.476(141.7)	140(87.5)	72.3(106.9)
	Blin	128(145)	40.483(165.6)	220(202.1)	81.3(141.8)
	pov	87(100)	28.060(141.7)	117.5(108.3)	25.7(106.9)
	pg2	93(88)	33.335(123.4)	132.5(83.3)	49.3(80.1)
	pg4	88(101)	31.131(141.7)	120(110.4)	39.4(106.9)
COMPAR	orig	3(3)	11.267(7.5)	0(0)	0(0)
	dup	9(7)	14.722(19.2)	200(133.3)	30.7(156)
	Blin	6(8)	13.516(24.8)	100(166.7)	20(230.7)
	pov	4(5)	11.417(13.6)	33.3(66.7)	1.3(81.3)
	pg2	6(5)	12.505(13.6)	100(66.7)	11(81.3)
	pg4	-	-	-	-
DEMUX38	orig	5(8)	12.078(7.0)	0(0)	0(0)
	dup	17(10)	19.131(13.1)	240(25)	58.4(87.1)
	Blin	21(14)	22.655(31.4)	320(75)	87.6(348.6)
	pov	10(12)	14.922(25.3)	100(50)	23.5(261.4)
	pg2	13(13)	16.971(19.2)	160(62.5)	40.5(174.3)
	pg4	7(12)	15.285(19.2)	40(50)	26.5(174.3)
DIS18SG	orig	66(26)	22.234(18.7)	0(0)	0(0)
	dup	179(57)	43.667(62.4)	171.2(119.2)	96.4(233.7)
	Blin	136(160)	41.910(98.0)	106.1(515.4)	88.5(424.1)
	pov	165(94)	27.786(49.7)	150(261.5)	25(165.8)
	pg2	124(99)	34.575(50.2)	87.9(280.8)	55.5(168.4)
	pg4	107(92)	31.288(37.5)	62.1(253.8)	40.7(100.5)

Table 1. continue

circuit		#slc(#mcl)	Max. Delay [ns]	Area Overhead (%)	Speed Decreasing (%)
MULTIPL	orig	14(10)	16.734(19.7)	0(0)	0(0)
	dup	38(29)	21.759(38.0)	171.4(190)	30(92.9)
	Blin	53(45)	25.342(68.5)	278.6(350)	51.4(247.7)
	pov	32(33)	20.396(56.3)	128.6(230)	21.9(185.8)
	pg2	33(37)	22.172(62.4)	135.7(270)	32.5(216.8)
	pg4	40(33)	20.492(44.1)	185.7(230)	22.5(123.9)
AVERAGE	orig	-	-	0(0)	0(0)
	dup	-	-	186.2(116.1)	59.1(141.4)
	Blin	-	-	216.4(286.7)	70.3(259.9)
	pov	-	-	99.1(126.6)	18.3(148.6)
	pg2	-	-	138(147.9)	39(138.6)
	pg4	-	-	108.8(195.3)	30.8(110.8)

On the average, the duplication scheme needed 186.2(116.1)% area overhead, the scheme with Bose-Lin code needed 216.4(286.7)% area overhead, and the scheme with parity prediction needed 99.1(126.6)% for pov, 138(147.9)% for pg2, and 108.8(195.3)% for pg4 of area overhead.

7 Conclusion

This paper describes a suitable approach for generating two-level combinational circuits with concurrent error detection based on duplication of function, parity-check codes and Bose-Lin codes. We describe an approach for insertion of concurrent error detection in synthesizable VHDL RTL description and then use a commercial synthesis tool to generate the implementation into FPGA or CPLD technology that allows easy tradeoff between overhead and fault coverage. An advantage of having a VHDL RTL description of the design is that it can be used for fast RTL simulation, and it provides a high-level, easy to understand documentation of the design's function. We insert the concurrent error detection circuitry at the RTL level rather than at the gate level because insertion at the front-end of the synthesis process has the following advantages:

a) The synthesis tool can take the error detection circuitry into account when satisfying timing constraints (as well as other constraints on power, testability, etc.)

b) Inserting the error detection circuitry at the RTL level can be easily and seamlessly incorporated into the standard flow.

References

[1] M. J. Ashjaee and S. M. Reddy, "On totally self-checking checkers for separable codes," *IEEE Trans. on Computers*, vol. C-26, pp. 737–744, Aug. 1977.

[2] P. K. Lala, *Self-checking and fault-tolerant digital system design*. San Francisco: Morgan Kaufman Publishers, 2001.

[3] N. Jha and S. J. Wang, "Design and synthesis of self-checking VLSI circuits," *IEEE Trans. on CAD*, vol. 12, pp. 878–887, June 1993.

[4] D. K. Pradhan and J. J. Stiffer, "Error correcting codes and self-checking circuits in fault tolerant computers," *IEEE Computer*, vol. 13, no. 3, pp. 27–37, Mar. 1980.

[5] J. E. Smith and G. Metze, "Strongly fault secure logic networks," *IEEE Trans. on Computers*, vol. C-27, pp. 491–499, June 1978.

[6] N. A. Touba and E. J. McCluskey, "Logic synthesis of multilevel circuits with concurrent error detection," *IEEE Transaction on CAD*, vol. 16, pp. 783–789, 1977.

[7] K. De and et al., "RSYN: A system for automated synthesis of reliable multilevel circuits," *IEEE Trans. on VLSI systems*, vol. 2, pp. 186–195, June 1994.

[8] N. Jha, "Totaly self-checking checker design for Bose-Lin, Bose, and Blaum codes," *IEEE Trans. on CAD*, vol. 10, pp. 136–143, Jan. 1991.

[9] Y. Tohma, "Coding techniques in fault-tolerant, self-checking, and fail-safe circuits," in *Fault-Tolerant Computing: Theory and Techniques, Volume 1*, Pradhan, Ed. New Jersey: Prentice-Hall, 1986, pp. 336–415.

[10] B. W. Johnson, *Design and Analysis of Fault Tolerant Systems*. Reading, MA, USA: Addison Wesley, 1990.

[11] D. Das and N. A. Touba, "Synthesis of circuits with low-cost concurrent error detection based on Bose-Lin codes," *Journal on Electronic Testing: Theory and Applications (JETTA)*, vol. 15, pp. 145–155, Aug. 1999.

[12] N. A. Touba and E. McCluskey, "Logic synthesis techniques for reduced area implementation of multilevel circuits with concurrent error detection," in *Proc. of ACM/IEEE Int. Conference on Computer-Aided Design (ICCAD)*, San Jose, CA, Nov. 1994, pp. 651–654.

[13] K. Mohanram *et al.*, "A methodology for automated insertion of concurrent error detection hardware in synthesizable Verilog RTL," in *Proc. of IEEE International Symposium on Circuits and Systems*, Scottsdale, Arizona, USA, 2002, pp. 577–580.

[14] J. C. Lo *et al.*, "An SFS Berger check prediction ALU and its application to self-checking processor designs," *IEEE Trans. on Computer Aided-Design*, vol. 11, pp. 525–540, Apr. 1992.