Synthesis of Circuits with Low-Cost Concurrent Error Detection Based on Bose-Lin Codes

Debaleena Das and Nur A. Touba

Computer Engineering Research Center
Department of Electrical and Computer Engineering
University of Texas, Austin, TX 78712-1084
E-Mail: {ddas, touba}@ece.utexas.edu

Abstract

This paper presents a procedure for synthesizing multilevel circuits with concurrent error detection based on Bose-Lin codes. Bose-Lin codes are an efficient solution for providing concurrent error detection as they are separable codes and have a fixed number of check bits, independent of the number of information bits. Furthermore, Bose-Lin code checkers have a simple structure as they are based on modulo operations. Procedures are described for synthesizing circuits in a way that their structure ensures that all single-point faults can only cause errors that are detected by a Bose-Lin code. This paper also presents an efficient scheme for concurrent error detection in sequential circuits. Both the state bits and the output bits are encoded with a Bose-Lin code and their checking is combined such that one checker suffices. Results indicate low area overhead. The cost of concurrent error detection is reduced significantly compared to other methods.

1. Introduction

In applications where dependability and data integrity are important, concurrent error detection (CED) circuitry is used to detect transient and intermittent errors. Early detection of errors is crucial for preserving the state of the system and preventing data corruption. The move towards deep-submicron technologies with lower voltage levels and smaller noise margins is increasing the susceptibility of systems to transient and intermittent faults thereby making CED increasingly important. This paper presents automated procedures for synthesizing both combinational and sequential circuits with low-cost CED circuitry. The CED circuitry can detect all on-line errors due to single-point faults as well as enhance off-line testability and reduce BIST overhead [Nicolaidis 89], [Gupta 96].

One general approach for CED is to encode the outputs of a circuit with an error detecting code and have a checker that monitors the outputs and gives an error indication if a non-codeword occurs (as illustrated in Fig. 1).

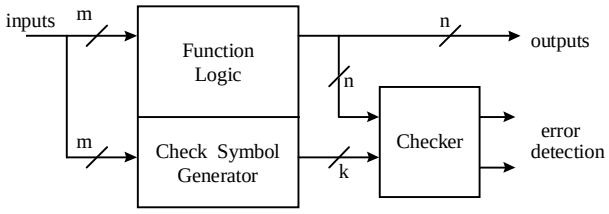

Figure 1. General Structure of a Self-Checking Circuit

To detect all single-point faults, the error detecting code that is used must be such that all possible errors in the circuit due to single-point faults result in non-codeword outputs. Efficient CED schemes exist for circuits with regular structures such as PLA's [Mak 82], [Nicolaidis 91] and ALU's [Pradhan 86], [Lo 92], [Gorshe 96]. However, for arbitrary multilevel combinational and sequential logic circuits more efficient approaches for synthesizing circuits with CED circuitry are needed [Gossel 93].

One approach that has been proposed for synthesizing arbitrary multilevel circuits with concurrent error detection is based on a Berger code. A Berger code is an optimal code for detecting all unidirectional errors [Berger 61]. Jha and Wang [Jha 93] proposed a synthesis method in which the circuit is synthesized using only algebraic transformations (such as those used in the algebraic script in MIS [Brayton 87]) such that the resulting circuit can be transformed so that it is *inverter-free*, i.e., has inverters only at the primary inputs (PI's). The primary outputs (PO's) are then encoded with a Berger code. Since the inverters are only at the PI's, any error caused by an internal fault (i.e., fault not at the

PI's) will produce a unidirectional error at the POs and therefore is guaranteed to be detected by the Berger code. Note that for any concurrent error detection scheme (including duplication), detection of faults at the primary inputs cannot be guaranteed unless encoded inputs are used, however, if the inputs to the circuit are outputs of another concurrently checked logic block, then the only undetectable PI faults are break faults after the checker [Khodadad-Mostashiry 80]. De, *et al.*, [De 94], described a constrained technology mapping procedure that maintains the inverter-free property during technology mapping thereby fully automating the synthesis method based on a Berger code. In [De 94], results are presented showing that a significant reduction in overhead is achieved by using the Berger code scheme compared with using duplication.

A Bose-Lin code [Bose 85] is an optimal systematic code for detecting up to t unidirectional errors and requires fewer check bits and a much simpler checker than a Berger code. In [Gorshe 96], it was shown that a significant hardware reduction could be achieved by using a Bose-Lin code instead of a Berger code for designing an ALU with CED. In this paper, a method is presented for synthesizing arbitrary multilevel circuits with CED based on a Bose-Lin code. The circuit is synthesized in a way that its structure ensures that all internal faults can only cause errors that are detected by a Bose-Lin code. Thus, a simpler Bose-Lin code can be used in place of a Berger code resulting in significant hardware reduction without loss of error detection capability.

Another contribution of this paper is a new and very efficient scheme for CED in sequential circuits. A sequential circuit with CED was proposed in [Jha 93]. The state bits are encoded with an m-out-of-n code and the output bits are encoded with a Berger code. Separate checkers are required to check the state bits and the output bits. In this paper, both the state bits and the output bits are encoded with a Bose-Lin code. An efficient technique is used to combine the checking the state bits and the outputs bits together with a single checker thereby reducing overhead. Moreover, since a Bose-Lin code is a separable code, no constraints are placed on the state encoding. Thus, it is compatible with conventional state assignment tools and also applicable to sequential circuits where some of the flip-flops may be storing data. By using the proposed procedure instead of a duplicate-and-compare approach, the cost of CED is significantly reduced while still detecting all single-point faults. The paper is organized as follows: Section 2 explains Bose-Lin codes. Section 3 presents the synthesis procedure. Sections 4 and 5 give the results for combinational and sequential circuits respectively. Section 6 concludes the paper.

2. Bose-Lin Codes

Bose-Lin codes are optimal codes for detecting up to t unidirectional errors. These codes were developed by Bose and Lin [Bose 85]. They are systematic 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.

The codes are constructed by counting the number of ones or zeroes, similar to the Berger codes. The counts are then modified depending on t. For t = 2 and 3, the counts are performed modulo 4 and 8, resulting in 2 and 3 check bits, respectively. Bose-Lin codes with check bits greater than 3 are explained in detail in [Gorshe 96]. The synthesis technique proposed in this paper can be used to implement self-checking circuits for any t unidirectional error detecting Bose-Lin codes.

A totally self-checking checker design for Bose-Lin codes has been proposed in [Jha 91]. This checker is based on the Berger code checker proposed in [Marouf 78]. Modulo 2^r addition is implemented by discarding the appropriate MSBs. Table 1 gives the comparison of literal counts of a double error detecting Bose-Lin code checker [Jha 91] as compared to a Berger code checker [Marouf 78].

Table 1. Literal Count Comparison of Bose-Lin Code Checker versus Berger Code Checker

# of Bits	Berger [Marouf 78]	Bose-Lin [Jha 91]
8	94	65
16	199	133
24	303	221
32	408	290

3. Synthesis Procedure

The circuit synthesis must be done such that all internal faults can only cause errors that can be detected by a Bose-Lin code. 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 (this constraint is the

same as for a Berger code). The second constraint is that no non-PI node in the circuit can have a path to more than *t* PO's if a *t* unidirectional error detecting Bose-Lin code is used. These two constraints ensure that all errors due to internal faults can cause no more than *t* unidirectional errors and thus all errors will be detected by the Bose-Lin code.

Multilevel logic optimization improves circuit area by using operations that restructure and minimize the logic. As was described in [Jha 93], multilevel logic optimization with algebraic transformations (such as those used in the algebraic script in MIS [Brayton 87]) will result in a circuit that can be transformed so that it has inverters only at the PI's. So if the synthesis is restricted to algebraic transformations, then the inverter-free constraint can be satisfied.

The constraint that no non-PI node can have a path to more than *t* PO's can be satisfied by restricting fanout in the circuit. Multilevel logic optimization operations that restructure the circuit must be constrained so that they do not introduce fanout that creates a path from some node to more than *t* PO's. There are two restructuring operations that introduce fanout, resubstitution and extraction [Brayton 90]. When performing those two operations, a check must be made to ensure that any new fanout being introduced in the circuit does not violate the constraints.

In *resubstitution* one logic expression *f* is substituted into another logic expression *g* to reduce the literal count. Resubstitution introduces fanout in the circuit as *f* will fanout to *g*. A check must be made to see if a particular resubstitution operation will create a path from some node to more than *t* PO's. If so, then the resubstitution operation is not permitted when synthesizing for a Bose-Lin code.

In *extraction*, a common subexpression is factored out from a set of logic expressions. Fanout is introduced from the subexpression to the logic expressions that is was factored out of. Again, a check must be made to see if a particular extraction operation will create a path from some node to more than *t* PO's. If so, then the extraction operation is not permitted.

We modified the MIS logic synthesis [Brayton 87] system for use in synthesizing circuits with CED based on Bose-Lin Codes. MIS uses various filters to reduce the pairs of logic expressions that are considered for resubstitution (because there are so many of them). We added an additional filter to remove pairs of logic expressions for which adding fanout would create a path from some node to more than *t* PO's. We also modified the extraction routines in MIS so that they only factor out common subexpressions between logic expressions for which the fanout constraints are satisfied. We added a command to MIS which automatically adds the Bose-Lin check bit functions. This modified version of MIS can be used to synthesize circuits with CED based on any *t* unidirectional error detecting Bose-Lin code. The original circuit is simply read into MIS and the Bose-Lin check bit functions are added, and then the circuit is optimized using the algebraic script with the modified resubstitution and extraction commands. Provided a technology mapping procedure that preserves the structure of the circuit is used, e.g., tree-mapping [Keutzer 87], [Detjens 87], all errors due to internal faults are guaranteed to be detected.

A special class of circuits with CED that satisfies certain properties are called *self-checking circuits* [Anderson 71]. Self-checking circuits guarantee detection of the first error that occurs due to a fault in a specified fault class. If a totally self-checking Bose-Lin code checker is used, the implementation that is generated by our synthesis procedure is *self-checking* for internal single-point faults.

4. Self-Checking Combinational Circuits

Self-checking combinational circuits have been studied in [De 94]. De, *et al.* have shown that a significant overhead reduction can be achieved by using Berger codes as compared to duplication. In this paper, we have synthesized several benchmark circuits with CED based on double error detecting Bose-Lin codes (i.e., $t = 2$). Since we used double error detecting codes, there are always two check bits which are added to the output bits. The check bits indicate the number of output bits that have a logical value of '1' modulo 4. Synthesis is done as explained in Sec. 3. The synthesis procedure ensures that no node in the circuit can fanout to more than two PO's. Table 2 gives information about the benchmark circuits that were used (for comparison, we used the same set of benchmark circuits that were used in [De 94]).

Table 2 also gives the literal counts for the original circuit, the circuit with CED based on Berger codes as reported in [De 94], and our circuit with CED based on Bose-Lin codes. For most of the circuits the number of literals for CED based on Bose-Lin codes is significantly reduced compared to the Berger code case.

The advantages of the Bose-Lin code lies in a lower number of check bits and a simpler checker. The structural constraint that no node can fan out to more than two primary outputs can cause an increase in the number of literals in the CED circuit as is seen in *luc* and

Table 2. Literal Count Comparison for Berger versus Bose-Lin Code

Circuit				Berger Code			Bose-Lin Code		
Names	#inputs	#outputs	Lits in Functional Circuit	Lits in CED Circuit	Lits in Berger Checker	Total Number of Lits	Lits in CED Circuit	Lits in Bose-Lin Checker	Total Number of Lits
apla	10	12	171	444	146	**590**	343	114	**457**
br1	12	8	118	229	94	**323**	248	74	**322**
bw	5	28	200	187	355	**542**	304	274	**578**
b10	15	11	401	875	133	**1008**	671	108	**779**
clip	9	5	130	675	54	**729**	473	48	**521**
dc1	4	7	49	51	80	**131**	66	68	**134**
dc2	8	7	118	271	80	**351**	283	68	**351**
inc	7	9	134	230	107	**337**	188	88	**276**
in0	15	11	449	890	133	**1023**	766	108	**874**
luc	8	27	215	248	342	**590**	477	268	**745**
m1	6	12	86	190	146	**336**	100	114	**214**
p82	5	14	109	158	172	**330**	162	134	**296**
sa02	10	4	166	273	41	**314**	229	34	**263**
vg2	25	8	87	943	94	**1037**	327	74	**401**
wim	4	7	41	73	80	**153**	61	68	**129**
x6dn	39	5	335	668	54	**722**	648	48	**696**
5xp1	7	10	119	284	120	**404**	280	94	**374**

p82. For *p82*, the total number of literals for CED based on Bose-Lin codes is less due to a simpler checker. For *luc*, the total number of literals is better for the Berger code case. It should be noted here that the constraint that no node can fanout to more than two primary outputs will reduce routing complexity and therefore lead to less layout area.

Layout was done for some circuits having a reasonably large number of outputs - *bw, luc, m1* and *p82*. The results are presented in Table 3. The layout was done using the tools from Alliance. Layout in [De 94] was done using Timberwolf. Thus, comparison of the two methods is done on the basis of the percentage overheads with the original circuit. Table 3 clearly indicates that the cost of CED is considerably reduced by the use of Bose-Lin codes. Benchmark circuits *bw* and *luc* have less layout area overhead for CED based on Bose-Lin codes, even though the literal count for the Bose-Lin code case was more than that for the Berger code case. The best results are seen for the *bw* circuit as it has the largest number of outputs, 28. In general, the cost of CED based on Bose-Lin codes will be *much less* compared to other methods for circuits having a *large number of outputs*.

Table 3. Layout Area Comparisons

Circuit		Duplication		Berger Code			Bose-Lin Code	
Name	Area	Area	%Ovrhd	Orig. Area	Area	%Ovrhd	Area	%Ovrhd
bw	301104	1286256	**327**	742000	2520000	**240**	461448	**53**
luc	367488	1420434	**286**	756000	2259888	**199**	842634	**129**
m1	113520	341694	**200**	584584	1018584	**74**	136080	**20**
p82	149160	562248	**277**	436728	1145256	**162**	241488	**62**

5. Self-Checking Sequential Circuits

Here we propose a cost-effective design for sequential circuits based on Bose-Lin codes that has a number of attractive features. The general form of a sequential circuit with CED as proposed in [Jha 93] is shown in Fig. 2. The state bits are encoded with an *m*-out-of-*n* code and the output bits are encoded with a Berger code. The state bits and the output bits are checked separately and the two checker outputs are then combined with a final two rail checker (TRC) module.

Any single-point fault in the combinational part of the circuit can affect the outputs of the next state logic and/or the outputs of the output logic. The fault will be detected by the *m*-out-of-*n* checker and/or the Berger code checker respectively. However, faults in the flip-flops cannot be detected by this design. Our design ensures that all faults in the flip-flops are also detected.

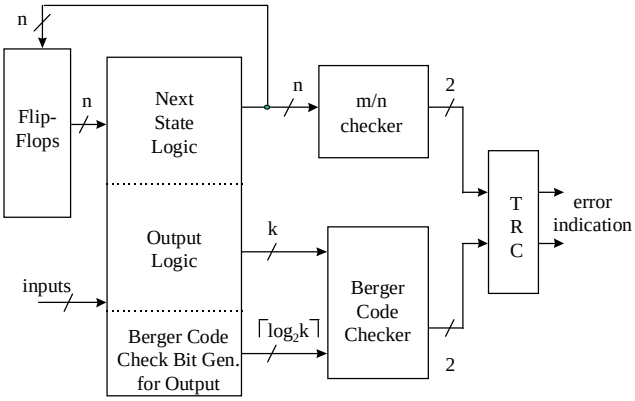

Figure 2. Self -Checking Sequential Machine Proposed in [Jha 93]

In our approach, we use Bose-Lin codes to encode both the state bits and the output bits. An efficient technique is used to combine the checking of the state bits and the output bits using a single checker (as illustrated in Fig. 3). We have used the following naming convention: *NS* (Next State bits) denotes the state bits generated from the next state logic block. *PS* (Present State bits) denotes the state bits after the flip-flops *i.e.* they are the *NS* after one clock cycle. NS_C are the two check bits for the next state logic. After passing through the flip-flop stage, they become check bits for the present state and are denoted as PS_C. Finally, Z_C are the check bits that encode both the output logic as well as the *PS*.

NS_C bits are computed in the usual manner by computing the number of ones modulo 4 in *NS*. The check bits for the output logic (Z_C), however, are computed as the number of ones modulo 4 in the ***output bits and PS***. The check bits can be thought of as encoding both the input and the output space of the circuit. Thus, only one checker is sufficient to check both the spaces. The Bose-Lin code checker has to be slightly modified for this application. The output bits are fed to the checker and the number of ones modulo 4 are computed. PS_C is added to the result of this computation and then the result is compared with Z_C. This arrangement ensures that the state bits and the output bits are checked together and all faults in the flip-flops are also detected.

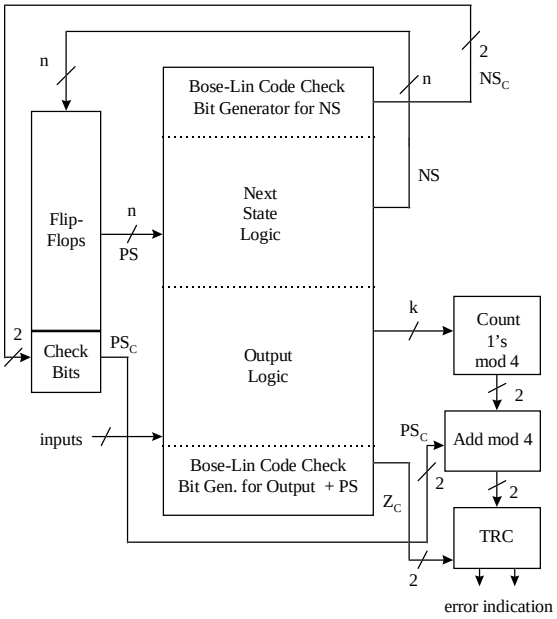

Figure 3. Proposed Self-Checking Sequential Machine

Analysis of fault detection: Our synthesis technique ensures that a non-PI node does not have a path to more than two POs. Thus, a single-point fault can result in errors in the following bit combinations: two *NS* bits, two output bits, one *NS* bit and one output bit, or a subset of any of the previous three. Errors in the *NS* bits will be clocked into the flip-flops. In the next cycle they are propagated to the *PS* bits and detected by the checker. Errors in the output bits will be detected in the same clock cycle as the occurrence of the error. Note that faults in the flip-flops are also detected since the checker checks the *PS* rather than *NS*. Even though the flip-flops may fanout to more than two PO's, single errors in the flip-flops are equivalent to being in a non-codeword state which is guaranteed to cause a mismatch with the PS_C check bits.

Table 4. Literal Count Comparison for Berger Code with *m*-hot state assignment versus Bose-Lin Code with Minimal State Assignment

Circuit	# inputs	# outputs	# states	*m*-hot Assign + Berger Code			Min. State Assign. + Bose-Lin		
				Lits in CED ckt.	Lits in checker+FF	Total # of literals	Lits in CED ckt.	Lits in checker+FF	Total # of literals
dk14	3	5	7	235	81	**316**	204	77	**281**
dk15	3	5	4	169	61	**230**	122	74	**196**
dk16	2	3	27	457	94	**551**	496	63	**559**
planet	7	19	48	810	314	**1124**	883	226	**1109**
styr	9	10	30	952	182	**1134**	1120	129	**1249**

Table 4 gives the comparison of the proposed method with [Jha 93]. The table gives the details of these circuits. State assignment has been done using MUSTANG [Devadas 88]. The number of flip-flops for the smaller circuits is the minimal number required to represent the states. Values of *m* for the *m*-hot encoding have been taken such that the number of flip-flops is comparable in both schemes.

Table 4 shows that the number of literals in the circuit with CED based on Bose-Lin codes is less for small circuits and larger for bigger circuits as compared to CED based on Berger codes. This is due mostly to the fact that an *m*-hot encoding gives better literal count results than MUSTANG for machines with a large number of states [Jha 93]. In actuality, comparing our literal counts with those in [Jha 93] is not a good comparison because the state assignment in the two cases is so different. We show the comparison just to illustrate that even with a less optimal state assignment (from the perspective of minimizing the literal count), we still get better results in most cases. If we were able to compare with the exact same state assignment we believe that our overhead would be significantly better. It should be noted here that it is not always possible to encode the states using *m*-hot codes. Bose-Lin codes, on the other hand, are separable codes and thus no constraints are placed on the state encoding. Thus, this scheme of CED is compatible with MUSTANG and any other state assignment tool. Also, it is applicable to general sequential circuits where some of the flip-flops may be storing data.

Table 4 also indicates that despite a larger number of literals in the circuit with CED, the total number of literals is comparable for both methods. The saving in cost of CED results from the smaller number of check bits for the Bose-Lin codes and the other saving in literals by our checker scheme. The modification to the normal Bose-Lin checker in our sequential circuit CED scheme is the modulo 4 addition of PS_C to the number of ones modulo 4 of the output bits. This requires 14 literals. Since the Bose-Lin code checker requires fewer literals than the Berger code checker, and we are using only one checker, the total number of literals in our proposed checker is much less than that of Jha and Wang [Jha 93] for a reasonably large number of outputs.

6. Conclusions

It has been found that Bose-Lin codes significantly reduce the cost of concurrent error detection for both combinational and sequential circuits. Synthesis and layout of combinational benchmark circuits with CED based on Bose-Lin codes show low hardware overheads, making it a practical solution. An efficient scheme for CED in sequential circuits has been proposed. This scheme can be extended to handle CED based on Berger codes. The advantages of this scheme are the following:

1) Only one checker is required for checking both the state bits and the output bits.
2) No constraints are placed on state encoding.
3) Faults in the flip-flops are guaranteed to be detected.

The synthesis procedure outlined in this paper was implemented in MIS, but can be implemented in other logic synthesis systems. Note also that the scheme described in this paper can be used to convert an existing sequential circuit (with both control state *and* data flip-flops) that does not have CED into one with CED by simply eliminating a sufficient number of fanouts to guarantee that all errors due to internal faults will be detected.

Acknowledgments

The authors would like to thank Zheng Yuan for her help in generating experimental results. This work is part of the TOPS project at the Center for Reliable Computing at Stanford University and was supported in part by the Defense Advanced Research Projects Agency (DARPA) under prime contract No. DABT63-94-C-0045.

References

[Anderson 71] Anderson, D.A., "Design of Self-Checking Digital Networks Using Coding Techniques," *Technical Report R-527*, Coordinated Science Laboratory, University of Illinois, Urbana, IL, 1971.

[Berger 61] Berger, J.M., "A Note on Error Detecting Codes for Asymmetric Channels," *Information and Control*, Vol. 4, pp. 68-73, Mar. 1961.

[Bose 85] Bose, B., and D.J. Lin, "Systematic Unidirectional Error-Detecting Codes," *IEEE Trans. on Computer Aided-Design*, Vol. C-34, No. 11, pp. 1024-1032, Nov. 1985.

[Brayton 87] Brayton, R.K., R. Rudell, A. Sangiovanni-Vincentelli, and A.R. Wang, "MIS: A Multiple-Level Logic Optimization System," *IEEE Trans. on Computer Aided-Design*, Vol. 6, pp. 1062-1081, Nov. 1987.

[De 94] De, K., C. Natarajan, D. Nair, and P. Banerjee, "RSYN: A System for Automated Synthesis of Reliable Multilevel Circuits," *IEEE Trans. VLSI Systems*, pp. 186-195, Jun. 1994.

[Detjens 87] Detjens, E., G. Gannot, R. Rudell, A. Sangiovanni-Vincentelli, and A. Wang, "Technology Mapping in MIS," Proc. IEEE/ACM Int. Conf. Computer-Aided Design (ICCAD), 1987, pp. 116-119.

[Devadas 88] Devadas, S., Ma, Hi-Keung, Newton, A.R., and A. Sangiovanni-Vincentelli, "MUSTANG: State Assignment of Finite State Machines Targeting Multilevel Logic Implementations," *IEEE Trans. on Computer Aided-Design*, Vol. 7, No. 12, pp. 1290-1299, Dec. 1988.

[Gossel 93] Gossel, M., and S. Graf, *Error Detection circuits*, London, NY: McGraw-Hill, 1993.

[Gorshe 96] Gorshe, S., and B. Bose, "A Self-Checking ALU Design with Efficient Codes," *Proc. of VLSI Test Symposium*, pp. 157-161, 1996.

[Gupta 96] Gupta, S.K., and D.K. Pradhan, "Can Concurrent Checkers help BIST?," *Proc. IEEE Int. Test Conf.*, 1992, pp. 140-150

[Jha 91] Jha, N.K., "Totally Self-Checking Checker Designs for Bose-Lin, Bose and Blaum Codes," *IEEE Trans. Computer-Aided Design*, Vol. 10, No. 1, pp. 136-143, Jan 1991.

[Jha 93] Jha, N.K., and S.Wang, "Design and Synthesis of Self-Checking VLSI Circuits," *IEEE Trans. Computer-Aided Design*, Vol. 12, No.6, pp. 878-887, Jun. 1993.

[Keutzer 87] Keutzer, "Dagon: Technology Binding and Local Optimization by DAG Matching," *Proc. IEEE/ACM 24th Design Automation Conf.*, 1987, pp. 341-347.

[Khodadad-Mostashiry 80] Khodadad-Mostashiry B., "Break Faults in Circuits with Parity Prediction," *Technical Note No. 183*, Center for Reliable Computing, Stanford University, Stanford, CA, Dec. 1980.

[Lo 92] Lo, J.-C., S. Thanawastein, and M. Nicolaidis, "An SFS Berger Check Prediction ALU and Its Application to Self-Checking Processor Designs," *IEEE Trans. on Computer Aided-Design*, Vol. 11, No. 4, pp. 525-540, Apr. 1992.

[Mak 82] Mak, G.P., J.A. Abraham, and E.S. Davidson, "The Design of PLAs with Concurrent Error Detection," *Proc. FTCS*, pp. 303-310, Jun. 1982.

[Marouf 78] M.A. Marouf and A.D. Friedman, "Design of Self-Checking Checkers for Berger Codes," *Proc. FTCS*, pp. 179-184, Jun. 1978.

[Nicolaidis 89] Nicolaidis, M., "Self-Exercising Checkers for Unified Built-In Self-Test (UBIST)," *IEEE Trans. on Computer Aided-Design*, Vol. 8, No. 3, pp. 203-218, Mar. 1989.

[Nicolaidis 91] Nicolaidis, M., and M. Boudjit, "New Implmentations, Tools, and Experiments for Decreasing Self-Checking PLAs Area Overhead," *Proc. of International Conference on Computer Design*, pp. 275-281, 1991.

[Pradhan 86] Pradhan, D.K., *Fault Tolerant Computing: Theory and Techniques*, Vol. 1, Englewood Cliffs, NJ: Prentice-Hall, 1986, Chap. 5.