Synthesis of Non-Intrusive Concurrent Error Detection Using an Even Error Detecting Function

Avijit Dutta and Nur A. Touba

Computer Engineering Research Center
Department of Electrical and Computer Engineering
University of Texas, Austin, TX 78712

Abstract

A new method for synthesizing non-intrusive concurrent error detection (CED) circuitry is presented. The idea is to use single-bit parity to detect all errors affecting an odd number of bits and then synthesize a circuit to detect the even errors. A novel statistical sampling and expanding methodology is proposed for constructing the even error detection circuitry. A major feature of the proposed methodology is that it allows very efficient tradeoffs between error coverage and overhead. While CED schemes that use a fixed checker based on a particular error detecting code are not amenable to simplification without a major impact on coverage, the proposed scheme can easily facilitate significant reductions in overhead with only a small loss in coverage. Experimental results show that the proposed scheme can provide very high levels of soft error protection at a fraction of the cost of duplication.

1. Introduction

When ionizing radiation from high-energy neutrons and alpha particles strike a sensitive region in a semiconductor device, they generate a dense track of electron-hole pairs that may be collected by a p-n junction resulting in a very short duration pulse of current causing a *single-event upset (SEU)* in the signal value. An SEU may cause a bit flip in some latch or memory element thereby altering the state of the system resulting in a *soft error*. Additionally, an SEU may occur in an internal node of combinational logic and subsequently propagate to and be captured in a latch. As process technology scales well below 100 nanometers, the higher operating frequencies, lower voltage levels, and smaller noise margins make integrated circuits increasingly susceptible to SEUs resulting in a dramatic increase in soft errors. Studies indicate that the soft error failure rate will become unacceptable even in mainstream commercial applications [Ziegler 96], [Cohen 99].

One way to detect soft errors is to use concurrent error detection (CED) circuitry that monitors the outputs of a circuit for the occurrence of an error [Gössel 93], [Nicolaidis 98]. If an error is detected, then the system can recover thereby preventing a failure. Detecting errors in logic circuits is much more expensive than in memories. While CED can be efficiently incorporated in memories due to their regular structure, logic circuits with their irregular structure present a much greater challenge. It is projected that in systems where memory CED is employed, soft errors in logic circuits will be the limiting factor for system reliability as technology continues to scale [Shivakumar 02], [Bowman 03, 04]. This paper focuses on the problem of providing CED in logic circuits.

The simplest CED scheme for logic circuits is to use duplication where the circuit is duplicated and the outputs are compared with an equality checker. While this is very simple to implement and provides very high error coverage, it requires over 100% overhead. A lot of research has been done on alternate schemes that are still applicable to any logic circuit but require less hardware overhead than duplication.

One class of techniques uses time redundancy. Multiple sampling of the outputs has been proposed in [Franco 94], [Metra 98], [Nicolaidis 99], [Favalli 02]. Self-dual functions have been proposed in [Saposhnikov 96, 98a]. These approaches have low hardware costs, but reduce performance.

Another class of techniques involves re-synthesizing the functional logic so that it has a more regular structure such that simple error detecting codes can be used to provide high coverage. Techniques have been developed for parity codes [De 94], [Touba 97], [Bolchini 97]; Berger codes [Jha 93], [Saposhnikov 98b]; and Bose-Lin codes [Das 99]. In cases where it is not desirable to re-synthesize the functional logic (e.g., cores, macrocells, handcrafted designs, legacy designs, etc.), these techniques are not applicable.

A third class of techniques uses non-intrusive CED where the functional logic is not modified. As shown in [Gössel 93], this problem can be formulated as follows (see illustration in Fig. 1). For a functional circuit with n inputs, $A=a_i,\dots,a_n$, and m outputs $Z=z_i,\dots,z_m$, let $EDF(a_i,\dots,a_n,\ z_i,\dots,z_m)$ be the error detecting function which is a Boolean function that is equal to 0 if the output vector Z is error-free, equal to 1 if the output vector Z has an error due to a fault in the specified fault class, and

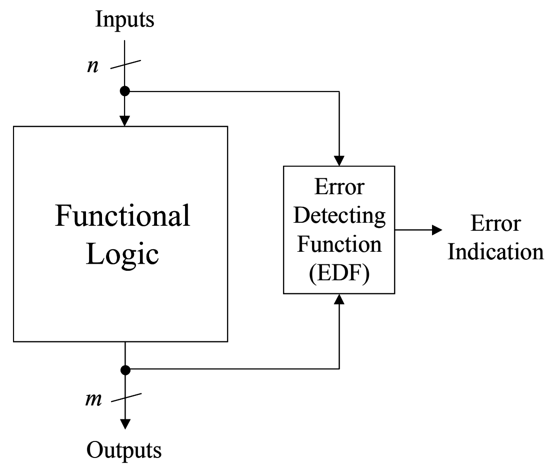

Figure 1. Non-Intrusive CED

equal to X (don't care) in all other cases (i.e., for input vector *A*, no fault can cause the output vector to be equal to *Z*). Any implementation of the Boolean function *EDF* will detect all errors due to the specified fault class. As pointed out in [Almukhaizim 04a], the *EDF* could be passed directly to a synthesis tool to produce the CED circuitry and if the synthesis algorithm could search exhaustively, it could find the optimal non-intrusive CED circuit. However, synthesis tools use heuristics to search the large space of solutions and consequently may obtain a sub-optimal solution. In fact the nature of the *EDF* function makes it particularly hard for synthesis tools to handle as it has a very large don't care space and many exclusive-or (XOR) factors which most synthesis tools are not good at finding. Thus, passing the *EDF* directly to a synthesis tool generally does not produce good results as shown in [Almukhaizim 04a]. Rather than trying to directly synthesize the *EDF*, researchers have explored structured implementations for the *EDF*. The basic approach for this is to place a compaction circuit at the outputs of the function logic to reduce them from *m* down to *k* and then synthesize a prediction circuit that independently predicts the *k* outputs. This is illustrated in Figure 2. One approach for compacting the outputs is to use a parity code which XORs together different subsets of the outputs [Sogomonyan 74], [Fujiwara 87]. If the parity code is selected so that no errors are masked, then 100% coverage can be maintained. In [Almukhaizim 04b], it was observed that the overhead for using a parity code is dominated by the prediction logic and a method based on entropy was proposed to guide the selection of the parity code to minimize the prediction logic. A technique for selecting the parity code with bounded error masking was described in [Tarnick 94]. In [Almukhaizim 04a], a more general design methodology that is not limited to parity was described for synthesizing the compaction circuit to ensure no error masking. In [Mohanram 03], CED based on a parity code is selectively disabled for some input vectors to tradeoff less coverage for less overhead in the prediction logic. In [Morozov 00], a technique for using a Berger code was described.

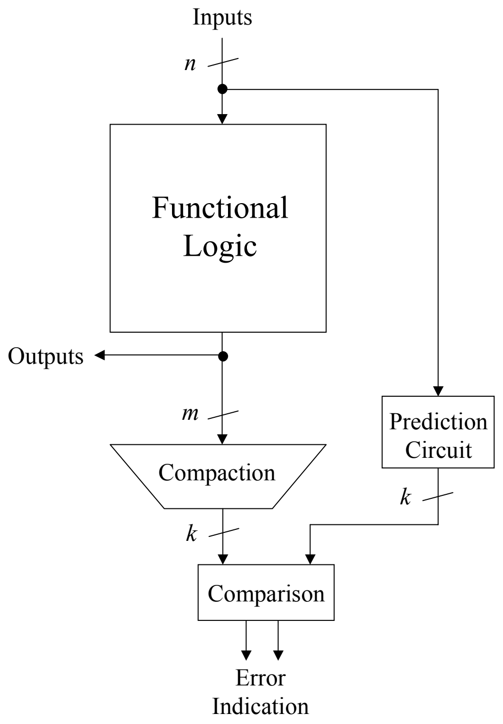

Figure 2. Basic Approach for Structured Implementation of Non-Intrusive CED

In this paper, a new method for synthesizing non-intrusive CED circuitry is presented. The idea is to use single-bit parity to detect all errors affecting an odd number of bits and then synthesize a circuit to detect the even errors. The key concept behind this approach is that most of the errors in the *EDF* function are single-bit errors. By using single-bit parity, all of the odd errors in the *EDF* function (which includes the single bit errors) become don't cares leaving only the even errors. The smaller number of even errors in the *EDF* function can be efficiently synthesized with most synthesis tools. In effect, the proposed method forces a decomposition of the *EDF* function in which the odd errors are covered with a single parity function and the even errors are covered via conventional logic synthesis with don't cares. Forming the *EDF* function for even errors by exhaustive simulation of all input vectors and all faults can be done only for small circuits. In order to handle larger circuits, a novel statistical sampling and expanding methodology is proposed. While most CED schemes use a fixed checker structure based on an error detecting code that it not amenable to simplification without a significant impact on error coverage. One of the nice features of the proposed scheme is that it provides very easy and efficient tradeoffs between coverage and overhead. A systematic approach is described for simplifying the even error detecting function that results in large reductions in overhead with only a minor loss in error coverage.

The paper is organized as follows: Sec. 2 provides an overview of the proposed scheme and its architecture. Sec. 3 describes the procedure for forming the even error detecting function. Sec. 4 explains how the proposed scheme allows for very efficient tradeoffs in coverage versus overhead. Experimental results are presented in Sec. 5. Section 6 concludes the paper.

2. Overview of Proposed Scheme

The proposed scheme involves combining single-bit parity with an even error detecting circuit. A block diagram for the proposed approach is shown in Fig. 3. The even error detecting circuit generates a two-bit error indication signal which normally has opposite values in the fault-free case and indicates an error by having equal values. An XOR-tree is used to compute the parity of the outputs of the functional logic. The parity predictor circuit predicts the complement of the parity of the outputs such that its output together with the XOR-tree output forms a two-bit error indication. The two pairs of error indication signals are then merged using a two-rail checker.

To simplify things, the even error detection function (*EVEN_EDF*) will be described in the rest of the paper as a single output function. The process of converting it so that it produces a two-bit error indication signal is trivial. It can be done by simply extracting one XOR factor, inverting it, and making it a separate output (i.e., extract any factor *E2* such that $EVEN\_EDF = E1 \oplus E2$ and use *E1* and *E2'* as outputs with the XOR gate removed). Thus, anytime *EVEN_EDF* is a 1, *E1* and *E2'* will have equal values indicating an error, and anytime *EVEN_EDF* is a 0, *E1* and *E2'* will have opposite values which is the normal error-free case.

Synthesizing the parity predictor circuit is exactly the same as for previously proposed methods. Synthesizing the even error detecting circuit is done by forming the *EVEN_EDF* function and giving it to a synthesis tool to synthesize. The challenge is how to form the *EVEN_EDF* function and that is the subject of the next section.

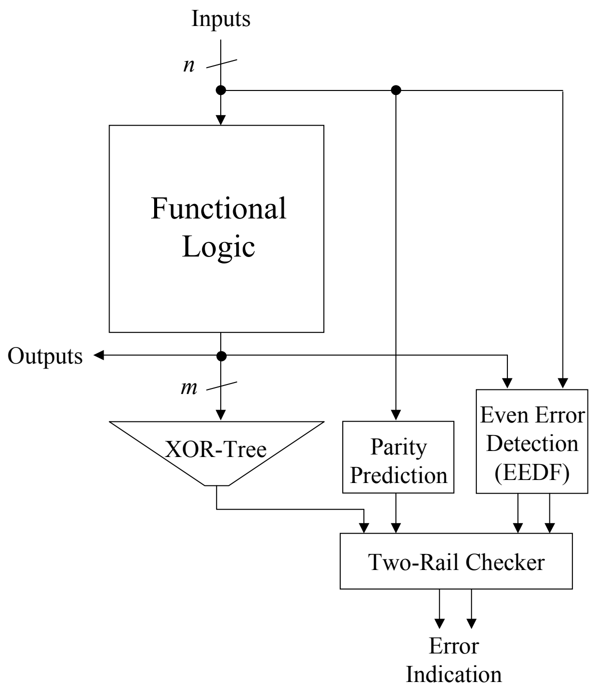

Figure 3. Block Diagram of Proposed Scheme

3. Forming *EVEN_EDF* Function

Given a functional logic circuit *F* with *n* inputs and *m* outputs, the simplest way to get the complete *EVEN_EDF* function that provides 100% coverage of all errors would be to exhaustively simulate *F* for all input vectors and faults. For each input vector, each fault is injected and the corresponding faulty output vector is obtained. If the faulty output vector has an even number of errors, then the minterm corresponding to the input vector and faulty output vector pair would be added to the ON-set of the *EVEN_EDF* function. This would continue until the complete ON-set for *EVEN_EDF* is formed. The OFF-set for *EVEN_EDF* is described by the functional logic circuit *F* itself. The DC-set includes anything that is not in the ON-set or OFF-set.

Forming the exact *EVEN_EDF* function through exhaustive simulation is intractable for all but the smallest circuits. Thus a less computationally complex procedure needs to be used for forming the *EVEN_EDF* function which will not necessarily obtain the exact ON-set. The proposed method involves using statistical methods to approximate the ON-set. Fortunately, good results can still be obtained even if the exact ON-set is not known. If some extra minterms from the DC-set are included in the ON-set, there is no loss of coverage, but possibly the synthesis tool may not obtain as optimal of a result. If some minterms are missing from the ON-set, there may be some loss of coverage. If the approximate ON-set is reasonably close to the exact ON-set, the impact in terms of either the optimality of the synthesis or the coverage can be kept very small. Moreover, if one is interested in trading off less coverage for less overhead, this can be nicely facilitated by approximating the ON-set in a way that the missing minterms simplify the logic implementation. Note that nothing from the OFF-set can be included in the ON-set because then the error indication would give a false alarm. The proposed method avoids this by construction as will be seen.

The proposed method for approximating the ON-set of the *EVEN_EDF* function involves random sampling of the input space for each fault combined with a *bit-stripping* operation. The procedure is described as follows:

Input: Functional logic circuit *F*, fault list, and number of simulations to do per fault (*L*).

Output: Approximate ON-set for *EVEN_EDF* function.

Step 1: Prune fault list – All faults that have a structural path to only one output are pruned from the fault list as they will never cause even errors.

Step 2: Randomly simulate *L* input vectors for each fault in fault list – The value of *L* is a parameter for this procedure that allows tradeoffs between runtime versus accuracy.

Step 3: For any vector that causes an even error, perform bit-stripping – Select a bit in the input vector and flip its value to the opposite of its current value and fault simulate. If the error is no longer even, then the input bit is flipped back to its original value. Otherwise, the input bit is changed to an X since an even error occurs regardless of the value of that input bit. This process is repeated for all the bits in the input vector one by one. The order in which the bits are processed is selected randomly each time a new vector is processed so that no particular order is repeated. The purpose of bit-stripping is to convert the input vector into an input cube that covers a large set of minterms.

Step 4: Add to the ON-set each input cube obtained in step 3 along with its corresponding output cube – Each input cube found in step 3 is fault simulated to obtain its corresponding 3-valued output cube. Together they specify a cube of minterms that are added to the ON-set of the *EVEN_EDF* function.

The procedure above produces an approximation of the ON-set for the *EVEN_EDF* function. Rather than simulating all of the input vectors for each fault (which would be exponential), only *L* vectors are simulated per fault where *L* is a user-specified value based on the desired level of accuracy in approximating the ON-set. Each input vector that causes an even error is expanded into a cube using bit-stripping. The resulting cube after bit-stripping contains many input vectors that also cause an even error. Some input vectors that cause an even error may not be found using this procedure because they may not be contained in any of the input cubes generated through bit-stripping. The larger the value of *L*, the more input cubes that are generated per fault and hence the less chance of missing an input vector that causes an even error for the fault. Missing input vectors that cause even errors means the ON-set for the *EVEN_EDF* function will be missing minterms which may result in some loss of coverage. However, on the good side, the minterms that are included in the *EVEN_EDF* function are contained in cubes (due to the way they were generated) and thus may simplify the logic implementation of the approximate *EVEN_EDF* function compared to the exact *EVEN_EDF* function that gives 100% coverage. The other source of approximation in the procedure is that one output cube is associated with all the input vectors contained in an input cube. In reality, of course, each input vector corresponds to only a single output vector and not a whole cube of output vectors. While the output cube is guaranteed to contain the correct output vector, it also contains many other output vectors thereby resulting in extra minterms being placed in the ON-set which should actually be in the DC-set. There is no risk of any minterms from the OFF-set ending up in the ON-set since the output cube always contains an even error (this is ensured by the way the bit-stripping is done) and thus it can never contain a fault-free output vector. The fact that the approximate ON-set contains some minterms from the DC-set does not impact the coverage at all. Potentially it could make the logic implementation of the approximate EVEN_EDF function more complex compared with the exact EVEN_EDF, but the fact that the additional minterms in the ON-set are contained in cubes (due to the way they were generated) the impact generally will not be significant.

Since the procedure is based on statistical sampling, no special ATPG is required to construct the EVEN_EDF function.

4. Coverage versus Overhead Tradeoffs

Because the procedure described in Sec. 3 approximates the ON-set, there is no guarantee of what the final coverage will be. Fault injection experiments are required to determine what coverage is achieved with the proposed method. A simple approach for this would be to do a large number of simulations where stuck-at faults are randomly injected in the functional logic circuit and random patterns are simulated. If an odd error results, it is guaranteed to be detected by the parity network. If an even error results, then the even error detecting function is simulated to see if it detects it. The percentage of all faulty output vectors that are detected gives the coverage.

This simple approach treats all faults as equally likely. More accurate results could be obtained by using a more sophisticated modeling (e.g., the one described in [Alexandrescu 02]).

If the coverage is not high enough, the procedure described in Sec. 3 can be repeated with a larger value of L to obtain a more accurate approximation of the *EVEN_EDF* function and then the even error detecting circuit can re-synthesized.

If lower overhead is desired for the CED circuitry, a strategy for achieving this while minimizing the loss of coverage is as follows. When the input cubes are generated via bit-stripping in Step 3 of the procedure described in Sec. 3, a threshold can be set on the size of the cubes. If the size of the input cube is not larger than the threshold, then the input cube is simply discarded and not added to the ON-set. The reasoning behind this strategy is that small input cubes contain only a small number of input vectors while requiring a potentially large amount of logic to implement (depending on the extent to which they can be merged or factored with other cubes). By discarding these cubes, the impact on the overall coverage is minimal while the benefit in reducing overhead is substantial. This strategy can be very effective in trading off a small loss in coverage for a large reduction in overhead. This is one of the key advantages of the proposed schemes and will be highlighted in the experimental results.

5. Experimental Results

Experiments were performed on some MCNC benchmark circuits [Yang 91]. The area results for the circuits were obtained using Synopsis Design Analyzer with the precompiled HTO18.db (0.18 micron) technology library. The area reported is the cell area.

Table 1 compares the area overhead for the self-checking circuits implemented using the duplication method and the proposed scheme. Both are non-intrusive and hence do not require re-synthesis of the functional logic. The circuit information and the optimized area for the MCNC benchmark circuits with no CED can be found under the first major heading.

Under the second and third major headings the results corresponding to the duplication method and the proposed scheme are given, namely the area for the circuit with CED and the percentage area overhead compared with the optimized functional logic without CED. For the proposed scheme different tradeoffs between area overhead and coverage are shown. The last coverage/overhead entry for each circuit shows the case where no even error detecting circuit is used (i.e., where only single-bit parity is used). To increase coverage, the even errors have to be detected. With a sufficiently large value of L, 100% coverage was obtained for most circuits to give a reference point. Note that the percentage area overhead was computed as follows:

% overhead =((area with CED – optimized area without CED) / (optimized area without CED)) ×100

The coverage was computed in the manner described in Sec. 4 where faults were randomly injected in the functional logic and random patterns were simulated. The coverage is defined as the number of output vectors that contained errors that were detected by the CED. Of course, duplication always provides 100% coverage.

As can be seen from the results, significant reductions in area overhead can be achieved with relatively small reductions in coverage. It is interesting to note that in most cases, getting the last 1-2% of coverage is very expensive. By going from 100% down to 99-98% coverage, a significant reduction in the CED overhead can be achieved. The likely reason for this is that there are a number of hard to sensitize paths that lead to even errors. Since few patterns sensitize these paths, the probability of soft errors occurring along these paths is very small. However, detecting these soft errors requires a lot of hardware. This phenomenon is illustrated in Figs. 4-6 which are graphs of coverage versus overhead. As can be seen in these graphs, the CED hardware required to increase the coverage rises somewhat linearly until the coverage reaches the high 90's at which point a lot of hardware is required to detect the last few percent of soft errors. The proposed method provides a very efficient way to take advantage of this phenomenon by allowing the designer the option of reducing the CED overhead significantly with only small loss in coverage.

Table 1. Comparison of Proposed Method with Duplication

Circuit				Duplication		Proposed		
Name	Num. PI	Num. PO	Area	Area	Overhead (%)	Area	Overhead (%)	Coverage (%)
apla	10	12	5348	12298	130.2	8252	54.3	100
						7487	40.3	99.2
						6417	20.5	88
						6150	11.5	72.4
br1	12	8	2876	6614	129.5	3756	30.6	100
						3402	18.3	99.8
						3258	13.3	91.2
						3169	10.2	76.4
chkn	29	7	8120	18940	133.5	16516	103.4	99.8
						15687	93.2	98.7
						15509	91.4	96.7
						15395	89.6	90
dc2	8	7	3021	7046	133.2	4894	62.1	100
						3697	22.4	99.4
						3359	11.2	90.3
						3217	6.5	84.6
exp	8	18	4176	9396	131.3	5846	40.4	100
						5203	24.6	97.4
						5094	22.2	80
						4927	18.4	70.2
wim	4	7	1236	3414	176.2	2224	80.2	100
						2007	62.4	98.9
						1928	56.3	80.2
						1903	54.0	74.2
5xp1	7	10	2320	5220	125.0	4461	92.3	100
						4245	83.6	98
						4129	78.4	88
						4036	74.0	81
b12	15	9	1208	3020	150.1	2356	110.2	100
						2391	98.4	97.3
						2379	97.6	78.4
						2343	94.2	72.2
cu	14	11	1180	3068	160.2	2950	150.1	100
						2430	106.4	98.2
						2289	94.3	72
						2230	89.7	64.2
sao2	10	4	2124	4885	129.9	3557	67.5	100
						3241	52.6	97.8
						3160	48.8	82
						3107	46.3	71.3
misex1	8	7	849	1970	132.0	1867	120.2	99.8
						1793.1	112.1	99.2
						1528.4	80.2	84
						1446.2	70.4	76

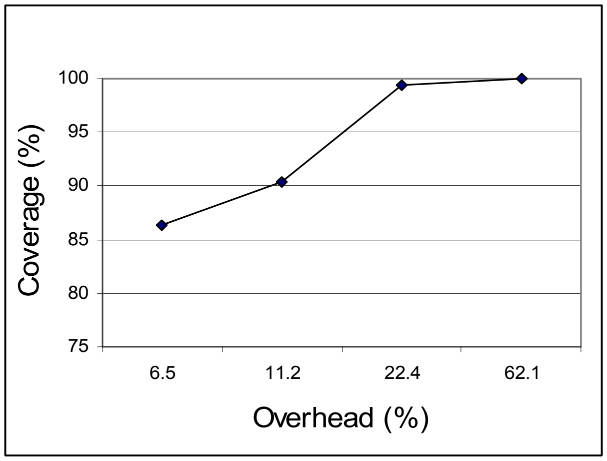

Figure 4. Coverage vs. Overhead for *dc2*

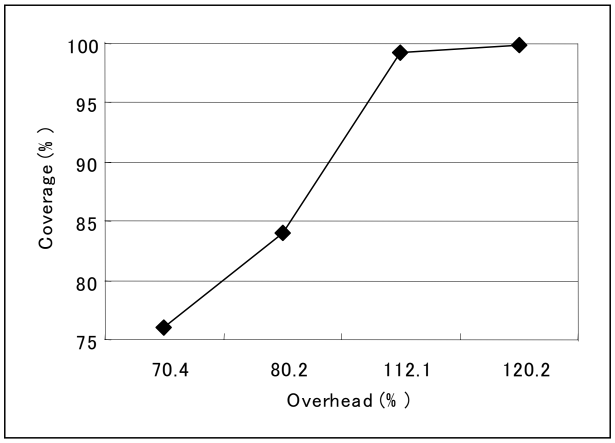

Figure 5. Coverage vs. Overhead for *misex1*

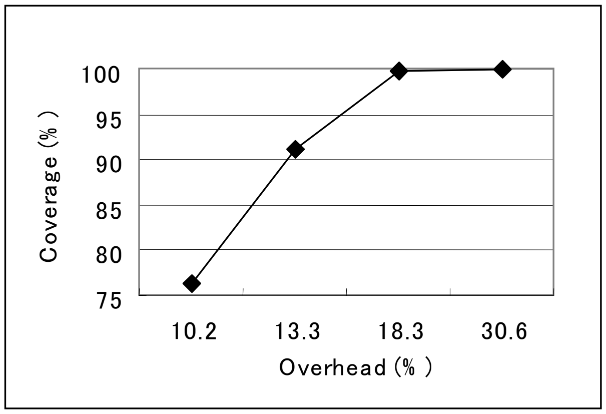

Figure 6. Coverage vs. Overhead for *br1*

6. Conclusions

The proposed method provides an efficient way to achieve high levels of soft error protection with reduced overhead. It is non-intrusive and thus does not require any modification to the functional logic itself.

Acknowledgements

This research was supported in part by the National Science Foundation under Grant No. CCR 0426608.

References

[Alexandrescu 02] Alexandrescu, D., L. Anghel, and M. Niholaidis, “New Methods for Evaluating the Impact of Single Event Transients in VDSM ICs,” *Prof. of Int. Symp. on Defect and Fault Tolerance*, pp. 99-107, 2002.

[Almukhaizim 04a] Almukhaizim, S., P. Drineas, and Y. Makris, "Concurrent Error Detection for Combinational and Sequential Logic via Output Compaction," *Proc. of Int. Symp. on Quality Electronic Design*, pp. 319-324, 2004.

[Almukhaizim 04b] Almukhaizim, S., P. Drineas, and Y. Makris, "Cost-Driven Selection of Parity Trees," *Proc. of VLSI Test Symposium*, pp. 319-324, 2004.

[Bowman 03] Bowman, R.C., “Technology scaling trends and accelerated testing for soft errors in commercial silicon devices”, *Proc. IEEE International On-Line Testing Symposium*, pp. 4, 2003.

[Bowman 04] Bowman, R.C., “Soft errors in commercial integrated circuits”, *International Journal of High Speed Computing*, Vol. 14, No.2, pp. 299-309, 2004.

[Bolchini 97] C. Bolchini, F. Salice, and D. Sciuto, “A Novel Methodology for Designing TSC Networks based on the Parity Bit Code,” *Proc. European Design and Test Conference*, pp. 440-444, 1997.

[Cohen 99] Cohen, N., *et al*., “Soft Error Considerations for Deep-Submicron CMOS Circuit Applications,” *International Electron Devices Meeting,* 1999.

[Das 99] Das, D., and N. A. Touba, “Synthesis of Circuits with Low-Cost Concurrent Error Detection Based on Bose-Lin Codes,” *Journal of Electronic Testing: Theory and Applications,* Vol. 15, Nos. 1/2, pp. 145-155, Aug. 1999.

[De 94] De, K., *et al.*, “RSYN: A System for Automated Synthesis of Reliable Multilevel Circuits,” *IEEE Trans. VLSI Systems*, pp. 186-195, Jun. 1994.

[Favalli 02] Favalli, M., and C. Metra, “Online Testing Approach for Very Deep-Submicron ICs,” *IEEE Design and Test of Computers*, Vol. 19, No. 2, pp. 16-23, Mar. 2002.

[Franco 94] Franco, P., and E. J. McCluskey, “On Line Delay Testing of Digital Circuits,” *Proc. VLSI Test Symposium*, pp. 167-173, 1994.

[Fujiwara 87] Fujiwara, E., and K. Matsuoka, “A Self-Checking Generalized Prediction Checker and Its Use for Built-In Testing,” *IEEE Trans. Computers*, Vol. C-36, No. 1, pp. 86-93, Jan. 1987.

[Gössel 93] Gössel, M., and S. Graf, *Error Detection Circuits*, McGraw-Hill Book Company, London, 1993.

[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.

[Mohanram 03a] Mohanram, K., E.S. Sogomonyan, M. Gössel, and N.A. Touba, "Synthesis of Low-Cost Parity-Based Partially Self-Checking Circuits," *Proc. of International On-Line Test Symposium*, pp. 35-40, 2003.

[Mohanram 03b] Mohanram, K., and N.A. Touba, "Cost-Effective Approach for Reducing Soft Error Failure Rate in Logic Circuits," *Proc. of International Test Conference*, pp. 893-901, 2003.

[Metra 98] Metra, C., M. Favalli, and B. Ricco, "Online Detection of Logic Errors Due to Crosstalk, Delay and Transient Faults," *Proc. International Test Conference*, pp. 524-533, 1998.

[Morozov 00] Morozov, A., V.V. Saposhnikov, Vl.V. Saposhnikov, and M. Gössel, "New Self-Checking Circuits by Use of Berger-Codes," Proc. of On-Line Testing Workshop, 2000, pp. 141-146, 2000.

[Nicolaidis 98] Nicolaidis, M., and Y. Zorian, "Online Testing for VLSI – A Compendium of Approaches," *Journal of Electronic Testing: Theory and Applications,* Vol. 12, Nos. 1/2, pp. 7-20, Feb. 1998.

[Nicolaidis 99] Nicolaidis, M., "Time Redundancy Based Soft-Error Tolerance to Rescue Nanometer Technologies," *Proc. of VLSI Test Symposium*, pp. 86-94, 1999.

[Saposhnikov 96] Saposhnikov, Vl.V., A. Dmitriev, M. Gössel, and V.V. Saposhnikov, "Self-Dual Parity Checking – A New Method On-Line Testing," Proc. of VLSI Test Symposium, pp. 162-168, 1996.

[Saposhnikov 98a] Saposhnikov, Vl.V., V.V. Saposhnikov, A. Dmitriev, and M. Gössel, "Self-Dual Duplication for Error Detection," *Proc. of Asian Test Symp.*, pp. 296-300, 1998.

[Saposhnikov 98b] Saposhnikov, V.V., *et al.*, "A New Design Methodology for Self-Checking Unidirectional Combinational Circuits," *Journal on Electronic Testing: Theory and Applications,* Vol. 12, Nos. 1/2, pp. 41-53, Feb. 1998.

[Shivakumar 02] Shivakumar, P., M. Kistler, S.W. Keckler, D. Burger, and L. Alvisi, "Modeling the Effect of Technology Trends on the Soft Error Rate of Combinational Logic," *Proc. International Conference on Dependable Systems and Networks*, pp. 389-398, 2002.

[Sogomonyan 74] Sogomonvan, E., "Design of Built-In Self-Checking Monitoring Circuits for Combinational Devices," Automation and Remote Control, Vol. 35, No. 2, pp. 280-289, 1974.

[Tarnick 94] Tarnick, S., "Bounding Error Masking in Linear Output Space Compression Schemes," *Proc. of Asian Test Symposium*, pp. 27-32, 1994.

[Touba 97] Touba, N.A., and E. J. McCluskey, "Logic Synthesis of Multilevel Circuits with Concurrent Error Detection," *IEEE Trans. on Computer-Aided Design,* Vol. 16, No. 7, pp. 783-789, Jul. 1997.

[Yang 91] Yang, S., "Logic Synthesis and Optimization benchmarks,Version 3.0, *Tech. Report*, Microelectronics Centre of North Carolina, 1991.

[Ziegler 96] Ziegler, J.F., *et al.*, "IBM Experiments in Soft Fails in Computer Electronics (1978-1994)," *IBM Journal of Research and Development*, Vol. 40, pp. 3-18, 1996.