SAT-based Code Synthesis for Fault-Secure Circuits

Dalirsani, Atefe; Kochte, Michael A.; Wunderlich, Hans-Joachim

Proceedings of the 16th IEEE Symp. Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFT'13) New York City, NY, USA, 2-4 October 2013

doi: http://dx.doi.org/10.1109/DFT.2013.6653580

Abstract: This paper presents a novel method for synthesizing fault-secure circuits based on parity codes over groups of circuit outputs. The fault-secure circuit is able to detect all errors resulting from combinational and transition faults at a single node. The original circuit is not modified. If the original circuit is non-redundant, the result is a totally self-checking circuit. At first, the method creates the minimum number of parity groups such that the effect of each fault is not masked in at least one parity group. To ensure fault-secureness, the obtained groups are split such that no fault leads to silent data corruption. This is performed by a formal Boolean satisfiability (SAT) based analysis. Since the proposed method reduces the number of required parity groups, the number of two-rail checkers and the complexity of the prediction logic required for fault-secureness decreases as well. Experimental results show that the area overhead is much less compared to duplication and less in comparison to previous methods for synthesis of totally self-checking circuits. Since the original circuit is not modified, the method can be applied for fixed hard macros and IP cores.

Preprint

SAT-based Code Synthesis for Fault-Secure Circuits

Atefe Dalirsani, Michael A. Kochte, Hans-Joachim Wunderlich
ITI, Universität Stuttgart, Pfaffenwaldring 47, D-70569 Stuttgart, Germany
Email: {dalirsani, kochte}@iti.uni-stuttgart.de, wu@informatik.uni-stuttgart.de

***Abstract*—This paper presents a novel method for synthesizing fault-secure circuits based on parity codes over groups of circuit outputs. The fault-secure circuit is able to detect all errors resulting from combinational and transition faults at a single node. The original circuit is not modified. If the original circuit is non-redundant, the result is a totally self-checking circuit.**

At first, the method creates the minimum number of parity groups such that the effect of each fault is not masked in at least one parity group. To ensure fault-secureness, the obtained groups are split such that no fault leads to silent data corruption. This is performed by a formal Boolean satisfiability (SAT) based analysis. Since the proposed method reduces the number of required parity groups, the number of two-rail checkers and the complexity of the prediction logic required for fault-secureness decreases as well. Experimental results show that the area overhead is much less compared to duplication and less in comparison to previous methods for synthesis of totally self-checking circuits. Since the original circuit is not modified, the method can be applied for fixed hard macros and IP cores.

***Index Terms*—Concurrent error detection (CED), error control coding, self-checking circuit, totally self-checking (TSC)**

I. Introduction

In safety critical applications, reliability and data integrity are of high importance, but may be compromised by latent defects, wear-out or transient faults such as crosstalk, power supply noise or radiation effects [1–3]. Concurrent error detection (CED) techniques are widely used to detect errors that appear during the operation of the circuit, irrespective of their permanent or transient nature [4].

One way to implement CED is to encode the outputs of a circuit with an error detecting code as presented in Fig. 1. The circuit (*functional logic*) generates the functional output bits. In the *checker*, check bits are computed from the output bits and a codeword is constructed. The checker compares the codeword to the one generated by the *prediction logic* and indicates an error in case of a mismatch.

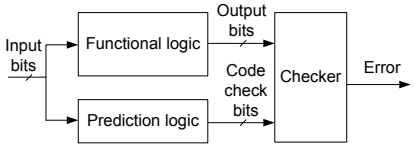

Fig. 1. General structure of CED with error detecting code

Duplication-with-comparison (DWC) is a conventional approach for CED of single- and multi-bit errors in which the duplicated circuit is considered as prediction logic. The outputs of the original and duplicated circuit (not necessarily the same implementation) are compared in the checker to signal an error when the outputs disagree. DWC incurs a relatively high area overhead of typically around 120% [5].

The theory of self-checking design classifies CED techniques [6] as follows: A circuit is *fault-secure* if for every fault of the target fault model the circuit never produces an incorrect codeword output for the code input. A circuit is *self-testing*, if for every fault it generates a non-codeword output for at least one code input. It is *Totally Self-Checking (TSC)* if it is self-testing and fault-secure. A non-redundant fault-secure circuit which receives all possible input patterns during operation is also TSC.

In self-checking design based on error detecting parity codes, the circuit outputs are distributed among one or more parity groups. A parity code (parity tree) computes the parity over the outputs of each group. Fault-secureness for all single stuck-at faults can be achieved by avoiding logic sharing between any two outputs in the same parity group, i.e. the outputs in one parity group are *structurally independent* [4]. One synthesis method is to use a single parity bit and prohibit logic sharing among the circuit outputs by replicating the shared logic [7, 8]. This solution causes a high area overhead. Therefore, many schemes distribute the outputs among multiple parity groups and the outputs of each group are encoded individually. Touba et al. [9] introduce a greedy algorithm which employs a cost function to find a parity code with potentially low area cost. The prediction logic is synthesized together with the original circuit under structural constraints so that outputs in the same parity group are structurally independent. In [10], the circuit is modified so that each fault is observable at an odd number of outputs for each input configuration.

Mitra et al. [5] provide a quantitative comparison of the overhead of various CED schemes and the protection against common-mode failures. Their investigation shows that duplicated systems sometimes have marginally higher area overhead compared to parity-based schemes while providing higher protection against common-mode failures. This altered the research direction of CED schemes based on error detecting codes to include extra functionalities: In [11], parity check codes have been used to extend CED to perform fault diagnosis and error correction. In [12], a 2-phase non-greedy algorithm of k-partitioning and local search is used to reduce the power consumption of a self-checking structure. In [13], the parity code search is guided by an entropy-based metrics to find low-power implementations. This technique requires the explicit computation of a large error detection table with entries for each fault and detecting input pattern.

Some techniques try to detect just as many errors as possible at lower overhead: In [14] and [15], only critical soft errors are targeted. The method in [16] is effective only at target faults with high sensitization probability. In [17], split-parity codes are investigated which detect all odd multibit errors with certainty and all even errors with a probability of 50%. When the circuit structure and functionality is known,

one may choose a function-inherent code checking method like [18] which exploits internal redundancies of the circuit.

In this paper, we present a novel approach for synthesizing parity prediction and checker logic for arbitrary logic circuits to ensure fault-secureness. The generated, extended circuit is even totally self-checking if the original one is non-redundant. The resulting self-checking structure uses multiple parity bits over the circuit outputs and is able to detect all errors resulting from single combinational and transition faults. It distributes the outputs among the parity groups such that for all testable faults in the circuit, the fault effect is not masked in at least one parity group. A Boolean satisfiability (SAT) based formal analysis assures the fault-secureness property. Since the proposed method reduces the number of required parity groups, the complexity of the parity prediction and checker logic decreases. The overall hardware overhead is less compared to previous self-checking approaches and for the first time, efficient fault-secureness is achieved for circuits with up to 15k gates.

The paper is organized as follows: Section II gives an overview of the proposed algorithm. Section III and IV explain the algorithm in detail. Section V discusses the results, followed by the conclusion.

II. Overview

The goal of this work is to synthesize checker and prediction logic to obtain a fault-secure circuit from a given circuit. With respect to a fault set F, a circuit is *fault-secure* [6] if and only if

$$\forall f \in F : \forall i \in I^n : C(i) = C_f(i) \vee P(C(i)) \neq P(C_f(i)). \quad (1)$$

I^n is the set of possible input vectors of n bits. $C(i)$ is the circuit response for input vector i in the fault free case, while $C_f(i)$ is the response under fault f. P computes the check bits over the circuit response $C(i)$ or $C_f(i)$.

Statement (1) declares that for any input vector, the fault either is not propagated to any circuit response ($C(i) = C_f(i)$), or it is detected by the check bits ($P(C(i)) \neq P(C_f(i))$). If a circuit is not fault-secure, there is at least one fault which causes *Silent Data Corruption (SDC)*, i.e. an undetected error in the circuit response. In parity codes, SDC occurs when all fault effects are masked in the parity trees. This happens if logic is shared in the transitive fanin of the inputs of a parity group and a multibit error of even magnitude occurs. Thus, the problem is to find an output partitioning such that the resulting parity group code ensures a fault-secure circuit according to statement (1).

To reduce the area overhead of a parity based fault-secure circuit, the number of parity groups should be minimized, and the parity group size should be increased as much as possible. This reduces the number of parity check bits as well as the size of the parity prediction and checker logic. If the maximum parity group size equals one, the resulting fault-secure circuit implements DWC.

Two circuit outputs are *dependent* if logic is shared in their input cones, and *independent* otherwise. Graph $G = (V, E)$ is constructed over the outputs V of the circuit. The edges E are the pairs of outputs which are dependent. The set of outputs which can belong to the same parity group without any risk of masking are independent and share no logic. They form an edgeless subgraph or independent set (the dual to a clique) in graph G.

A partitioning $P_k = \{G_i = (V_i, E_i) \mid 1 \leq i \leq k\}$ with edgeless subgraphs G_i and $\bigcup_i V_i = V$ corresponds to k parity groups over the outputs without logic sharing. Yet even logic sharing of outputs within a single group could be allowed as long as the fault effects are propagated to an odd number of outputs covered by another parity group. This relaxation increases the search space for an output partitioning and reduces the area overhead.

We propose a two step synthesis approach as shown in Fig. 2. Firstly we conduct a primary partition of the circuit outputs based on a topological analysis of the shared logic in the input cones of circuit outputs. The result is a minimum number of partitions k such that each fault effect is not masked in at least one parity group. This topological analysis cannot guarantee fault-secureness, but reduces the probability of silent data corruption and serves as starting point for the fault-secure synthesis step.

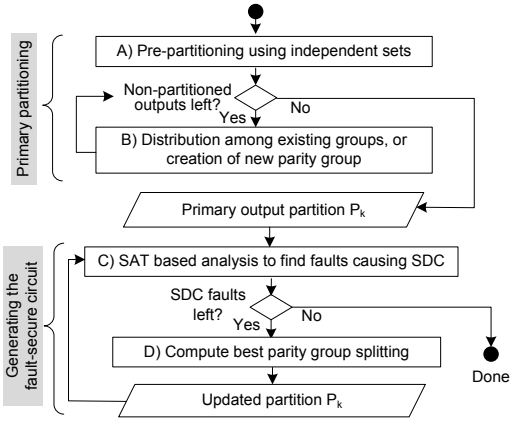

Fig. 2. Fault-secure circuit synthesis flow.

The second phase of the algorithm achieves fault-secureness by a SAT-based formal analysis to find the faults which may cause silent data corruption. Parity groups are split up such that the fault effect is observable in the resulting parity groups for all input patterns. This scheme is conducted iteratively until the circuit is fault-secure.

III. Primary Partitioning

The primary partitioning is achieved by a topological analysis of logic sharing in the input cones of the circuit outputs: When a gate g is in the input cone of only one output in group G_i, any combinational fault at g is propagated to at most one output and is not masked in parity group G_i. In this case, we say g is $covered$ by group or subgraph G_i:

$$Cov(g) = \begin{cases} 0 & g \text{ is not covered in any group} \\ 1 & g \text{ is covered in at least one group } G_i \end{cases}$$

In a circuit with l logic gates, the coverage of a partitioning P_k is defined as the number of covered gates: $Coverage(P_k) =$

$\sum_{i=1}^{l} Cov(g_i)$. Primary partitioning searches for a partition P_k such that $Coverage(P_k) = l$ with minimized k. As shown in Fig. 2, it consists of: (A) Computing a pre-partitioning constructed from independent sets of outputs in G, and (B) iteratively creating additional groups and distributing yet unpartitioned vertices among them.

A. Pre-Partitioning of Circuit Outputs

If some outputs are dependent, the partitioning needs at least two groups of outputs: $P_2 = \{G_1, G_2\}$. G_1 and G_2 are two independent sets of outputs in G which maximize $Coverage(P_2)$.

At first, the maximal independent sets for every pair of independent outputs are computed. Compared to the problem of listing all maximal independent sets [19], this constraint limits the runtime complexity to $\mathcal{O}(|V|^3)$.

For each independent set of outputs G_i, we compute the set of gates in the input cones (fanin) of all outputs in group G_i as: $Cone(G_i) = \bigcup_{v \in V_i} Cone(v)$, with $Cone(v)$ as the set of gates in the input cone of output v. As the outputs in G_i are independent, the number of covered gates in G_i equals $|Cone(G_i)|$. The independent set with maximum $|Cone(G_i)|$ is assigned to the first group of outputs $G_1 = (V_1, \emptyset)$.

The second group G_2 is similarly chosen from the list of independent sets of outputs, excluding the outputs in G_1. This increases the flexibility for adding the remaining unpartitioned vertices to the available groups.

B. Distribution of Unpartitioned Outputs

The yet unpartitioned outputs are assigned either to the groups so far established (two groups right after pre-partitioning), or to new groups if it is not possible to cover a gate in one of the existing groups: First, we assign as many outputs as possible to the available groups. All unpartitioned outputs share logic with the already partitioned ones in the groups G_i. However, we assign output v to group G_i, if all gates in the shared logic between $Cone(v)$ and $Cone(G_i)$ are covered in another group. In Fig. 3, for instance, v_5 shares logic with the input cone of v_3 in the striped area 'S'. v_5 is still assigned to group G_1 because the shared logic is also part of the input cone of v_2 and covered by G_2.

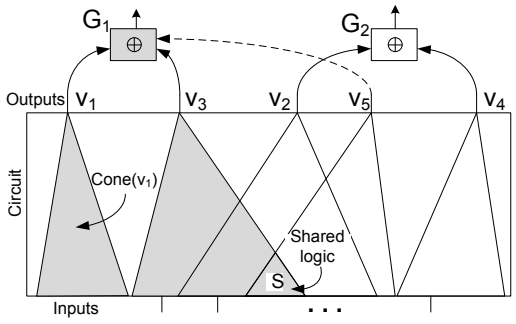

Fig. 3. Distribution of unpartitioned outputs

If an output v sharing gate g with group G_i is assigned to G_i, faults on gate g might be masked in G_i. This must be avoided when g is covered only by G_i. Thus, before each output assignment, we check the logic sharing of gates in $Cone(v)$ and $Cone(G_i)$ and allow the assignment of v to G_i only if no gates become uncovered. For instance, in Fig. 3, v_5 is not allowed to be added to G_2 because it shares logic with v_2 which is not covered in any other group.

To determine the order in which the outputs are assigned to the groups, we compute for each unassigned output its weight as:

$$w(v) = \max_{G_i \in P_k} \Delta Coverage(v, G_i),$$

where $\Delta Coverage(v, G_i)$ is the number of additionally covered gates when adding v to G_i. The output with highest weight is assigned first.

The distribution of outputs among established groups continues until no further assignment to the groups is possible. Then a new group is created. The new group is chosen among the independent sets of outputs excluding already partitioned outputs.

IV. Generating the Fault-Secure Circuit

Primary partitioning creates a small number of parity groups as starting point for the fault-secure synthesis. Primary partitioning cannot guarantee fault-secureness yet, since it only ensures that a fault effect is not masked in *one* parity group. There may be input vectors which propagate the fault effect *only* to different groups where error masking is not inhibited. However, primary partitioning reduces the runtime of the formal analysis by decreasing the probability of such silent data corruption (SDC).

Further splitting of parity groups until no logic is shared any more between outputs in the same group would lead to a solution similar to [20]. In our scheme, as shown in Fig. 2, we perform (C) a formal analysis to check which faults cause SDC and (D) selectively split the parity groups only to prevent SDC. For a non-redundant circuit, the resulting circuit is even totally self-checking, since the proposed algorithm does not modify the original circuit and all errors are detected.

Computing whether fault f causes silent data corruption (SDC) is an NP-complete problem. Exhaustive fault simulation of all faults and all input vectors (like the error detectability table in [13]) can be conducted only for small circuits with very few inputs. For larger circuits, this approach becomes infeasible.

The directed search for one input vector that causes silent data corruption under fault f, or the proof that such a vector doesn't exist, can be mapped to a test pattern generation (ATPG) problem [21]. In [22], it is shown that such a check can also be conducted using Boolean satisfiability (SAT). We map the problem to a SAT instance as explained below.

A. SAT-based Analysis of Fault-Secureness

Fig. 4 illustrates the principle idea to check if a fault causes silent data corruption (SDC). When functional outputs of the circuit C and its faulty copy C_f disagree ① and the generated parity bits are equal ②, SDC occurs. A SAT instance is constructed which is satisfiable if and only if there is an input vector such that for a fault f the following formula is true:

$$\exists\, i \in I^n : C(i) \neq C_f(i) \wedge P(C(i)) = P(C_f(i)).$$

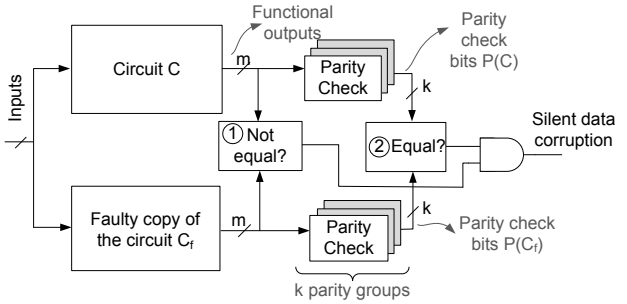

Fig. 4. Structure of the SAT instance to analyze fault-secureness.

A SAT instance is constructed for each fault. The instance models the gates in the original circuit C and the faulty copy C_f. The functional outputs of C and C_f are compared bitwise ①. The parity check structure computes the parity bits $P(C)$. $P(C)$ is compared with the check bits of the faulty circuit $P(C_f)$ in ②.

The SAT instance Φ is a Boolean formula represented as a union of clauses. The characteristic equations of the gates in the fault-free and faulty circuit C^{CNF}, C_f^{CNF} are obtained from the circuit netlist using the Tseitin transformation [23]. P^{CNF} and P_f^{CNF} are the clauses which are extracted from the parity check structure. The instance Φ is the conjunction or union of these clauses and the clauses required for comparison:

$$\Phi = C^{CNF} \wedge C_f^{CNF} \wedge \bigvee_{1 \le j \le m} (o^j \neq o_f^j) \wedge$$
$$P^{CNF} \wedge P_f^{CNF} \wedge \bigwedge_{1 \le j \le k} (p^j = p_f^j)$$

Φ is only satisfiable if the functional outputs (o^j) differ and the parity check bits (p^j) are equal. In this case, the satisfying model is an input vector which causes SDC.

B. Parity Group Splitting

After primary partitioning, the circuit outputs have been partitioned into k groups. The faults F^{SDC} which cause silent data corruption (SDC) have been determined by the presented formal analysis. For each fault $f \in F^{SDC}$, the SAT solver computed one input vector i_f causing SDC (although i_f may not be the only vector causing SDC under f). We simulate all faults $f \in F^{SDC}$ with input vector i_f to find the outputs and parity groups to which the fault effect is propagated.

Group splitting then refines the partitioning iteratively to achieve fault-secureness as shown in Fig. 5. The parity groups in which fault effects are masked and SDC occurs are analyzed as explained below.

Let V_f^{Gi} be the subset of outputs in group G_i to which the fault effect of f is propagated under input vector i_f. To avoid SDC of fault f under i_f in this group, the group must be split into at least two groups such that the fault effect is not masked any more.

There are $2^{(|V_f^{Gi}|-1)} - 1$ possibilities of splitting G_i. For each possibility, we compute by SAT analysis the number of faults for which SDC is avoided by the splitting. After computing the number of avoidable SDCs for all groups and all splittings, the group and splitting with highest reduction in

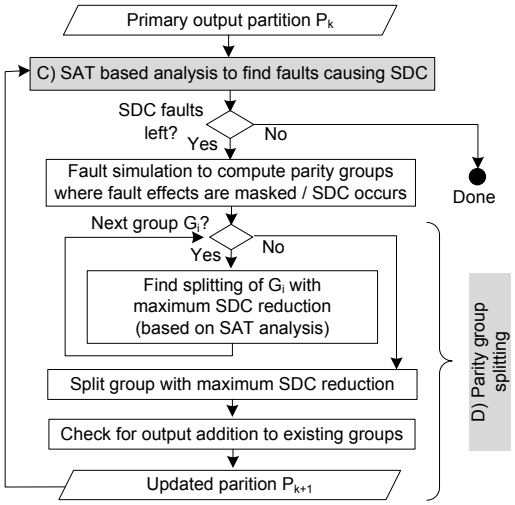

Fig. 5. Generating the fault-secure circuit.

SDC faults is constructed. The result is one additional parity group. Since new groups are created, a topological analysis is performed after each splitting to investigate the possibility of adding some outputs to available groups and avoid SDC that way. By this analysis, one output may be included in an additional group if it doesn't share any logic with the outputs in that group.

For large $|V_f^{Gi}|$, the explicit analysis of each splitting possibility is computationally too expensive. In this case, a topological analysis is used to divide G_i into two groups, one of them is the independent set of G_i, and the other has the minimum logic sharing in its input cones.

The process of SDC analysis and parity group splitting proceeds until no more faults cause silent data corruption. The parity prediction and checker logic are synthesized separately from the functional circuit. As the checker must be self-checking, we use cascadable two-rail checkers consisting of six logic gates [24] to construct a totally self-checking checker with k input pairs (for k parity groups). The checkers compare the check bits of the parity groups with the check bits computed by the prediction logic. The resulting circuit is fault-secure.

V. Experimental Results

We evaluate the area cost of the proposed method for benchmark circuits and compare the cost with previous work. In the circuits, output hold registers and prediction hold registers must be added [25] for the self-checking structure (Fig. 6) to avoid a longer critical path due to the checker logic.

A. Area Cost of Fault-Secure Synthesis

The area overhead of the fault-secure circuit consists of the parity prediction logic, the two-rail totally self-checking checker [24], the parity trees to generate the parity bits from functional outputs, and the hold registers as shown in Fig. 6.

The prediction logic and the checker are synthesized using Synopsys Design Compiler, optimized for area. The target library lsi10k is constrained to two-input gate primitives. The cell area of a two input NAND gate is 1 area unit.

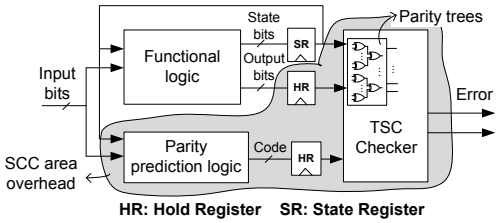

Fig. 6. Area overhead of the self-checking circuit (SCC).

Table I shows the results for the ISCAS'85 and ISCAS'89 circuits with more than 1000 gates (excluding multiplier c6288 for which efficient arithmetic coding techniques exist). Table II shows the results for the ITC'99 circuits. The first five columns list the circuit name, the number of inputs and outputs (primary and pseudo primary), the size in cell area units, and the number of collapsed faults.

Column 'DWC' reports the cell area overhead of the duplicated system, consisting of the copy of the circuit as predictor, hold registers and the checker.

The remainder of the tables reports the results of the proposed method. The number of parity groups after primary partitioning is given in column 'P.P. grps'. Column 'P.P. SDC' shows the number (percentage) of faults which cause silent data corruption after primary partitioning. Our investigation shows that after primary partitioning, for every testable fault in the circuit there exists at least one input pattern for which the functional outputs differ and a wrong parity bit is generated. Furthermore, we observe that typically only a low percentage of faults still cause SDC. This result emphasizes the importance of primary partitioning to reduce the effort of the formal fault-secureness analysis and group splitting.

The results of the final fault-secure circuits are given in columns 9 to 12: Column 'Parity bits' shows the number of groups that the outputs are distributed to. The next column gives the cell area overhead of the fault-secure circuit. For comparison with the duplication method, we list the *relative area cost* κ of the proposed method with respect to DWC:

$$\kappa = \frac{\text{Self-checking circuit area overhead}}{\text{Duplication-with-comparison (DWC) area overhead}} \quad (2)$$

For $\kappa < 1$, the proposed parity-based synthesis method incurs less area overhead than DWC, which is the case for all circuits. The area saving depends on the circuit size and structure. The weighted average of κ with regard to the circuit size over all ISCAS'89 benchmark circuits (up to 15K gates) is 0.56, i.e. only 56% of the area of DWC is required. To the best of our knowledge, this is the first method able to generate area-efficient fault-secure circuits for medium size circuits (up to 15K gates) in reasonable time. For most circuits, the runtime is less than a few minutes. For large circuits the runtime increases due to a higher number and more complex SAT checks. For s15850, the number of SAT checks increases by a factor of 3 compared to s13207 which causes higher runtime.

B. Comparison to Related Work

Most of the previous parity-based methods were applied to small size circuits only. We compare the area cost of the proposed method with the reported results of the parity-based self-checking methods in [5, 9]. In [5, 9], totally self-checking circuits are synthesized using multiple parity codes over groups of outputs, and replication of shared logic when required. The original circuit is modified. The experiments use the MCNC benchmark circuits which are relatively small with few outputs. For such circuits, parity based fault-secureness does not save much area compared to DWC [5], and DWC itself is not very expensive. Our method in average imposes a slightly lower overhead compared to [5, 9] even without requiring circuit modification (Due to space limit, the detailed result is omitted).

In [13], results for an error detection table and entropy based method are reported for MCNC and smaller ITC'99 circuits. We compare our results with the results for ITC'99 circuits with more than 100 gates. The relative cost κ' is computed according to eq. (2) using the area overhead of the self-checking circuit and the area overhead of duplication as reported in [13]. The last two columns of Table II list the number of parity bits and κ' of the method in [13]. The table also shows our results for much larger ITC'99 circuits which were not investigated in [13].

The weighted average of κ over the five circuits in the comparison with [13] is 0.51, i.e. 17% less than [13]. In addition, our method is also applicable to much larger circuits with an average value of κ of 0.58.

In average, our method outperforms the previous parity-based self-checking methods, without the need to modify the original circuit. The experiments demonstrate its applicability to circuits with up to 15k gates because of the combination of topological and formal techniques.

The proposed method incurs significantly less area overhead compared to DWC. For some circuits only half of the DWC area is required for fault-secureness with respect to all combinational and transition faults on a single node (including all stuck-at faults). This area saving needs to be traded off against the potentially higher coverage of multiple faults offered by DWC.

VI. Conclusion

This paper presented a novel synthesis method for fault-secure circuits without modification of the original circuit. It constructs the parity groups required to detect all errors resulting from single combinational and transition faults. During primary partitioning, a topological analysis distributes the outputs among as few parity groups as possible. A formal SAT-based analysis then assures fault-secureness by group splitting. Since the proposed method reduces the number of required parity groups, the number of two-rail checkers and the complexity of the prediction logic decreases as well. The results show that fault-secureness can be achieved with in average only 60% of the area overhead of duplication with comparison. The method also outperforms previous self-checking approaches regarding the area overhead.

Acknowledgement

Parts of this work were supported by the DFG under grants WU 245/10-2 (OTERA) and WU 245/12-1 (ROCK).

TABLE I. Results for ISCAS benchmark circuits.

Circuit					DWC	Proposed Method					
Name	PIs + PPIs	POs + PPOs	Cell area	Faults	Cell area overhead	P.P. grps.	P.P. SDC	Parity bits	Cell area overhead	**Relative cost** κ	Time (s)
c1908	33	25	1193	2100	1906	3	152 (19%)	10	1591	0.83	895
c2670	233	140	1819	3226	5867	4	248 (9%)	10	3173	0.54	538
c3540	50	22	2788	3332	3414	4	1395 (41%)	17	3309	0.97	14736
c5315	178	123	4274	6246	7829	8	685 (12%)	22	5708	0.73	9895
c7552	207	108	5078	8284	8198	6	842 (11%)	9	6119	0.75	3987
s953	45	52	786	1208	1818	6	127 (10%)	14	1020	0.56	154
s1196	32	32	1076	1386	1704	6	297 (21%)	22	1494	0.88	412
s1238	32	32	1108	1514	1736	5	263 (17%)	19	1463	0.84	494
s1423	91	79	1617	1680	2712	5	535 (31%)	21	1494	0.55	2881
s1488	14	25	1619	1710	2236	5	73 (4%)	13	1984	0.89	74
s1494	14	25	1629	1734	2246	5	73 (4%)	14	2015	0.90	79
s5378	214	228	5249	5474	8949	7	901 (16%)	41	5062	0.57	5342
s9234	247	250	9473	7760	13063	8	1622 (20%)	27	8380	0.64	4451
s13207	700	790	16179	12024	28368	6	1012 (8%)	21	12204	0.43	6394
s15850	611	684	17646	14016	27898	9	2957 (21%)	35	14289	0.51	122609

TABLE II. Results for ITC'99 benchmark circuits.

Circuit					DWC	Proposed Method						[13]	
Name	PIs + PPIs	POs + PPOs	Cell area	Faults	Cell area overhead	P.P. grps.	P.P. SDC	Parity bits	Cell area overhead	**Relative cost** κ	Time (s)	Parity bits	Relative cost κ'
b03	34	34	418	478	912	5	80 (16%)	10	408	0.45	95	4	0.83
b04	77	74	1381	1888	1803	4	256 (13%)	23	1140	0.63	636	-	-
b05	35	60	1708	2748	2892	5	966 (35%)	29	2241	0.77	6152	-	-
b06	11	15	140	176	419	3	11 (6%)	5	209	0.50	1	-	-
b07	50	57	893	1238	1750	5	347 (26%)	11	784	0.45	1144	6	0.58
b08	30	25	395	528	772	4	66 (12%)	12	499	0.65	12	5	0.60
b09	29	29	424	476	805	4	169 (35%)	6	322	0.40	56	7	0.40
b10	28	23	399	596	782	4	88 (14%)	12	551	0.70	12	7	0.67
b11	38	37	1180	1876	1745	6	223 (11%)	18	1346	0.77	355	-	-
b12	126	127	2394	3310	4129	7	449 (13%)	19	1861	0.45	457	-	-
b13	63	63	796	1002	1763	4	111 (11%)	14	734	0.42	94	-	-

REFERENCES

[1] S. Borkar, T. Karnik, and V. De, "Design and reliability challenges in nanometer technologies," in *Proc. ACM/IEEE Design Automation Conference*, 2004, pp. 75–75.

[2] S. Nassif, "Modeling and analysis of manufacturing variations," in *IEEE Conf. on Custom Integrated Circuits*, 2001, pp. 223–228.

[3] R. Baumann, "Soft errors in advanced computer systems," *IEEE Design & Test of Computers*, vol. 22, no. 3, pp. 258–266, may-june 2005.

[4] M. Goessel, V. Ocheretny *et al.*, *New Methods of Concurrent Checking*. Springer, 2008.

[5] S. Mitra and E. McCluskey, "Which concurrent error detection scheme to choose?" in *Proc. IEEE Int'l Test Conference*, 2000, pp. 985–994.

[6] M. Nicolaidis and Y. Zorian, "On-line testing for VLSI, a compendium of approaches," *Journal of Electronic Testing (JETTA)*, vol. 12, no. 1-2, pp. 7–20, Feb. 1998.

[7] K. De, C. Natarajan *et al.*, "Rsyn: a system for automated synthesis of reliable multilevel circuits," *IEEE Trans. VLSI Systems*, vol. 2, no. 2, pp. 186–195, june 1994.

[8] N. A. Touba and E. J. McCluskey, "Logic synthesis for concurrent error detection," Stanford, CA, USA, Tech. Rep., 1993.

[9] N. Touba and E. McCluskey, "Logic synthesis of multilevel circuits with concurrent error detection," *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, vol. 16, no. 7, pp. 783–789, jul 1997.

[10] C. Bolchini, F. Salice, and D. Sciuto, "Parity bit code: achieving a complete fault coverage in the design of TSC combinational networks," in *Proc. Seventh Great Lakes Symposium on VLSI*, 1997, pp. 32–37.

[11] S. Almukhaizim and Y. Makris, "Fault tolerant design of combinational and sequential logic based on a parity check code," in *Proc. IEEE Int'l Symp. on Defect and Fault Tolerance (DFT)*, 2003, pp. 563–570.

[12] S. Ghosh, N. Touba, and S. Basu, "Synthesis of low power CED circuits based on parity codes," in *Proc. IEEE VLSI Test Symposium*, 2005, pp. 315–320.

[13] S. Almukhaizim, P. Drineas, and Y. Makris, "Entropy-driven parity-tree selection for low-overhead concurrent error detection in finite state machines," *IEEE Trans. CAD*, vol. 25, no. 8, pp. 1547–1554, 2006.

[14] R. Vemu, A. Jas *et al.*, "A low-cost concurrent error detection technique for processor control logic," in *Proc. Design, Automation and Test in Europe (DATE)*, 2008, pp. 897–902.

[15] K. Mohanram and N. Touba, "Cost-effective approach for reducing soft error failure rate in logic circuits," in *Proc. IEEE International Test Conference*, 2003, pp. 893–901.

[16] K. Mohanram, E. Sogomonyan *et al.*, "Synthesis of low-cost parity-based partially self-checking circuits," in *Proc. 9th IEEE On-Line Testing Symposium (IOLTS)*, july 2003, pp. 35–40.

[17] M. Richter and M. Goessel, "Concurrent checking with split-parity codes," in *Proc. IEEE International On-Line Testing Symposium (IOLTS)*, 2009, pp. 159–163.

[18] C. Metra, D. Rossi *et al.*, "Function-inherent code checking: A new low cost on-line testing approach for high performance microprocessor control logic," in *IEEE Europ. Test Symposium*, 2008, pp. 171–176.

[19] E. Tomita, A. Tanaka, and H. Takahashi, "The worst-case time complexity for generating all maximal cliques and computational experiments," *Journal of Theoretical Computer Science*, vol. 363, no. 1, pp. 28–42, Oct. 2006.

[20] E. S. Sogomonyan and M. Goessel, "Design of self-testing and on-line fault detection combinational circuits with weakly independent outputs," *Journal of Electronic Testing (JETTA)*, vol. 4, pp. 267–281, 1993.

[21] M. Hunger and S. Hellebrand, "Verification and analysis of self-checking properties through ATPG," in *Proc. IEEE International On-Line Testing Symposium (IOLTS)*, 2008, pp. 25–30.

[22] G. Fey and R. Drechsler, "A basis for formal robustness checking," in *Proc. Int'l Symp. on Quality Electronic Design (ISQED)*, 2008, pp. 784–789.

[23] G. Tseitin, "On the complexity of derivation in propositional calculus," *Studies in constructive mathematics and mathematical logic*, vol. 2, no. 115-125, pp. 10–13, 1968.

[24] P. K. Lala, *Self-Checking and Fault-Tolerant Digital Design*. Morgan Kaufmann, 2001.

[25] C. Zeng, N. Saxena, and E. J. McCluskey, "Finite state machine synthesis with concurrent error detection," in *Proc. IEEE Int'l Test Conference*, 1999, pp. 672–679.