An Efficient Multiple-Parity Generator Design for On-Line Testing on FPGA

Petr Fišer, Pavel Kubalík, Hana Kubátová
Czech Technical University in Prague, Dept. of Computer Science and Engineering
e-mail: fiserp@fel.cvut.cz, xkubalik@fel.cvut.cz, kubatova@fel.cvut.cz

Abstract

We propose a method to efficiently design a "parity generator", which is a stand-alone block producing multiple parity bits of a given circuit. The parity generator is designed by duplicating the original circuit, XOR-ing given groups of its outputs and resynthesizing the whole circuit. The resulting circuitry is mostly smaller than the original circuit. The major task to be solved is to properly select the groups of outputs to be XORed to obtain multiple parity bits and maximally reduce the generator size. A method based on principles of the FC-Min minimizer is proposed in this paper. The parity generator is exploited in on-line diagnostics, to design self-checking circuits based on a modified duplex system.

1. Introduction

Systems realized by FPGAs are being more and more popular and widely used in many applications due to several advantages they possess, like a high flexibility in achieving multiple requirements such as cost, performance and turnaround time and the possibility of reconfiguration. The FPGA circuits can be used in mission critical applications such as aviation, medicine, railway applications, and space missions as well [1]. Most of FPGAs are based on SRAM memories sensitive to Single Even Upsets (SEUs), therefore a simple usage of FPGA circuits in mission critical applications without using any method of error detection (and possibly correction) is impossible. The Concurrent Error Detection (CED) techniques allow a faster detection of soft errors (errors which can be corrected by reconfiguration) caused by SEUs [2]. The probability of a SEU occurrence in the SRAM is described in [3].

The self-checking (SC) circuit based on a CED technique is used to detect an occurrence of a fault in the tested circuit. We use the Modified Duplex System (MDS) architecture [4]. The self checking circuit quality is determined by an area overhead and the number of undetectable faults while keeping dependability parameters [5].

This paper presents a parity generator design method based on parity bits grouping, by using the FC-Min minimizer [6, 7]. Here we exploit principles of sharing group implicants among two or more outputs of the function. The groups of outputs to be XORed are derived from the numbers of group implicants they share.

2. The Parity Generator

The self-checking circuit is constructed by duplicating the original circuit and XORing the outputs of the duplicate circuit, to obtain multiple parity bits. The code words obtained by the original circuit and the parity generator are then compared in the Checker, see Fig. 1.

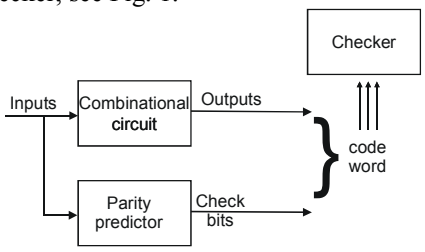

Figure 1: The self-checking circuit design

The number of used parity bits significantly influences the area overhead, together with the dependability parameters, see [8].

3. FC-Min

The output grouping method is based on the FC-Min minimizer principles [6]. The minimization is being conducted in a reverse way than the standard minimizers do. First, the group cover of the on-set of all output functions is found. After that the minimized implicants are produced by processing the source implicants, in order to satisfy the cover. Thus, group implicants are generated directly, not like in other minimization methods by reducing prime implicants of single functions. This approach makes FC-Min a very fast, since only implicants that will be a part of the final solution are produced.

The minimization process consists of two processes: the *Find Coverage* algorithm and *Implicants Generation*.

The Find Coverage process is the essential phase of FC-Min. The whole cover of the on-set of the multi-output function is found first, using the output part of the source function only. The algorithm tries to find a cover of the on-set by finding a rectangle cover of all the "1" values in the output matrix (description of the function's on-set).

An example of such a cover is shown in Fig. 2. There is shown a 5-input and 5-output function defined by 10 terms, in a form of a truth table. The result of the Find Coverage algorithm is a cover consisting of six *coverage elements*, $t_1 - t_6$. A coverage element is a Cartesian product of two sets, the *coverage set* $C(t_i)$ and the *coverage mask* $M(t_i)$. The coverage set describes the rows that are covered by t_i, the coverage mask gives the output variables covered by t_i. Our example coverage elements are shown in Tab. 1.

Each coverage element describes a potential implicant. For example, the *group* term (implicant) t_1 covers "1"s of the fourth and fifth output variable (y_3 and y_4) in vectors 4, 6 and 8.

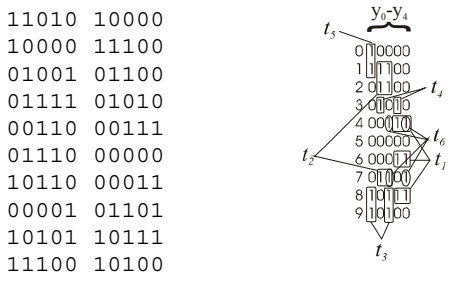

Figure 2: Cover of the output matrix

Table 1: Coverage elements from Fig. 2

Implicant	$C(t_i)$	$M(t_i)$
t_1	{4, 6, 8}	$\{y_3, y_4\} \equiv 00011$
t_2	{1, 2, 7}	$\{y_1, y_2\} \equiv 01100$
t_3	{8, 9}	$\{y_0, y_2\} \equiv 10100$
t_4	{3}	$\{y_1, y_3\} \equiv 01010$
t_5	{0, 1}	$\{y_0, y_1\} \equiv 10000$
t_6	{4, 7}	$\{y_2, y_4\} \equiv 00101$

After each coverage element is produced, it has to be validated, i.e., we must verify, whether there exist an implicant covering the "1"s in $C(t_i) \times M(t_i)$. This is done by directly generating the respective implicant. If this process fails, the coverage element is discarded and another, different one is computed.

Considering the conditions described above, a simple rule the implicants have to satisfy can be derived: the *minimum implicant* satisfying the particular cover can be constructed as a *minimum supercube* of all the input vectors corresponding to the rows of the cover of t_i. Moreover, this supercube must not intersect any term that is not included in the particular cover $C(t_i)$. In our example, the minimum implicant t_1 would be `(-01--)`.

4. The Output Grouping

The idea of grouping multiple-output function's outputs to form multiple parity bits is straightforward: we try to group together outputs having many *common group implicants*. Such outputs will more likely share some terms, thus grouping them together would be advantageous for a two-level implementation of the source multi-output function. We have found experimentally that the same effect can be observed for a multi-level synthesis as well; the outputs sharing many group implicants share a lot of logics in the multi-level implementation of the function as well [7]. When these outputs are connected by a XOR gate to form a parity bit and the circuit is resynthesized, the overall logic could be furthermore reduced.

Since there are often big numbers of possible group implicants (coverage elements) and output variables, it is not easy to combine the influences of the implicants. We have found that an efficient way to estimate the grouping of the outputs is by constructing a *grouping matrix G*. It is a symmetric matrix of dimensions [*m*, *m*], where *m* is the number of outputs. The value G[*i, j*] defines the "binding strength" of two output variables *i* and *j*.

The G matrix is being constructed during the coverage generation process. Firstly, the matrix is filled with zeros. After each valid coverage element is produced, the values in all the positions in G corresponding to all the couples of variables in $M(t_i)$ are increased by one. This describes an increased likelihood that the outputs y_3 and y_4 will be grouped together. For details see [7].

5. Experimental Results

5.1. The Overall Synthesis Process

The overall synthesis process, i.e., the way how all the tests have been performed will be described in this subsection.

The source functions for our experiments were the MCNC [11] benchmark circuits. The parity generator design process has been held in the following steps:

1. First, the benchmark described as a PLA structure has to be pre-processed, in order to generate the function's on-set and off-set, which is needed for FC-Min. This is done by ESPRESSO [10].
2. The circuit is then processed by FC-Min, to derive the output grouping.

3. The obtained groups of outputs are XORed, to obtain the parity bits. This is done by converting the original circuit's PLA into a BLIF [9] file by SIS [9] and appending the XOR gates to the outputs.
4. The obtained parity generator is decomposed into LUTs by SIS. The number LUTs is counted then, to make an estimation of the size of the parity generator.

5.2. The Efficiency of the Method

In order to evaluate the efficiency of the proposed method, we have compared the FC-Min based parity bits grouping with a purely random grouping. We have varied the number of parity bits from one to the number of the circuit's outputs. One limit, the 1-parity bit case, involves XORing all of the circuit's outputs, thus any "smart" output grouping method cannot come into effect. The second limit case, i.e., the number of parity bits equal to the number of outputs, corresponds to the original circuit (no XORs). For each benchmark circuit and a given number of parity bits, 500 random and 500 FC-Min based output groupings have been generated and the average of each was taken. A typical growth of the number of look-up tables (LUTs) for the FPGA realization obtained by SIS [9] is shown in Fig. 3a for a sqr6 MCNC [11] benchmark circuit. The size of the circuit grows with the number of parity bits here. It can be concluded that by XORing the circuit's output its size is reduced after resynthesis; producing the parity bits only is advantageous here, with respect to the total area.

On the other hand, Fig. 3b shows the alu2 benchmark results. Here the number of LUTs increases with decreasing the number of the parity bits, thus adding XORs to the circuit inputs involves the circuit size growth, even after the resynthesis. Adding XOR gates to their output just increases their complexity and, moreover, standard synthesis tools, like SIS are not able to handle such circuits efficiently [12]. Fortunately, such cases are quite rare.

Two curves are shown in figures 3a and 3b. One curve corresponds to the FC-Min based output grouping, one to a random grouping. We can see that the FC-Min grouping always produced a circuit having fewer LUTs.

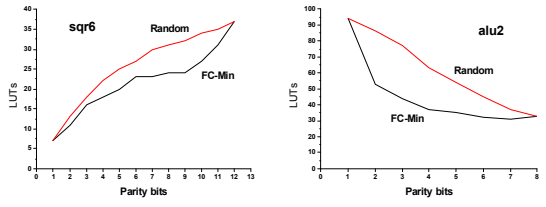

Figure 3a,b: The sqr6 and alu2 MCNC example

The summary results obtained from several MCNC benchmarks [11] are shown in Tab. 2. The "*Bench*" column shows the benchmark name, the number of its outputs follows (*m*). The ratio of the size of a 1-parity generator to the original circuit is shown in the next column ("*Ratio*"). 100% means no difference, circuits having the ratio less than 100% correspond to the Fig. 3a case, ratios higher than 100% to the Fig. 3b.

The average and maximum improvement obtained by our output grouping method, with respect to the random grouping is shown in the next two columns, in terms of the number of LUTs. The number of parity bits, where the maximum improvement was reached is indicated in the parentheses in the "*Max. impr.*" column. The average improvement values computed from results of 40 circuits are indicated in the last row.

Table 2: Output grouping results

Bench	*m*	*Ratio*	*Avg. impr.*	*Max. impr.*
alu1	8	3950.0%	11.7%	37.9% (4)
alu2	8	284.8%	25.3%	42.9% (3)
alu3	8	356.7%	3.5%	8.9% (4)
apla	12	32.2%	13.8%	28.0% (4)
br1	8	35.9%	7.4%	18.9% (5)
br2	8	15.7%	13.4%	36.4% (4)
in2	10	33.3%	6.1%	21.3% (7)
in7	10	86.2%	30.4%	51.5% (2)
m1	12	11.1%	6.8%	33.3% (4)
m2	16	13.3%	15.6%	32.4% (4)
mlp4	8	20.0%	6.6%	26.9% (6)
mp2d	14	151.4%	25.1%	42.0% (3)
newbyte	8	6.3%	11.7%	37.9% (4)
newcpla	16	52.4%	17.4%	28.2% (6)
newcpla2	10	19.4%	29.9%	53.3% (3)
p82	14	8.6%	4.3%	20.0% (10)
sex	14	47.4%	23.4%	38.1% (9)
sqr6	12	18.9%	13.9%	25.0% (9)
t4	8	192.9%	6.8%	28.6% (4)
tms	16	8.3%	10.1%	27.3% (10)
Average			10.1%	24.0%

5.3. The Dependability Parameters

Availability computations were used to compare our modified duplex system with a standard duplex system and with the TMR (Triple Modular Redundancy) system. Availability is a function of a time A(*t*), defined as the probability that a system is operating correctly and is available to perform its functions at an instant of a time *t*.

Dependability calculations are processed for a single parity first, then for a multiple parity. In the multiple parity case 2 or 3 parity groups were selected.

Our results of improved availability parameters are shown in Tab. 3. Here "*Bench*" is the name of the benchmark circuit, "*AO*" is the area overhead, "*FS*" is the probability that a fault is detected by a code word, "*ASS*" is the steady-state availability and "*Impr. ASS*" indicates the improvement of *ASS* against single parity when multiple parity is used. The results show that the multiple parity predictor technique can be used to improve steady state availability parameters. In some cases the improvement of availability parameter is more than 10%.

6. Conclusions

We have proposed an efficient method to design a multiple parity generator for on-line BIST. The method is based on properly choosing the original circuit's outputs to be XORed to obtain respective parity bits. The choice is being done by determining outputs that share many group implicants in the two-level representation of the multi-output function. These outputs share a lot of combinational logic and, most likely, the amount of the overall logic would be reduced by appending XOR gates to these outputs.

The availability parameters of the MDS architecture based on self-checking circuits have been calculated. The results show that using the multiple parity bits increase availability parameters at the price of a higher area overhead, with respect to the single parity case.

Acknowledgment

This research has been partially supported by MSMT under research program MSM6840770014 and FI-IM4/149 grant.

References

[1] D. Ratter, " FPGAs on Mars", www.xilinx.com,Xcell Journal Online, 2004.

[2] M. Bellato, P. Bernardi, D. Bortalato, et al., "Evaluating the effects of SEUs affecting the configuration memory of an SRAM-based FPGA", Design Automation Event for Electronic System in Europe 2004, pp. 584-589.

[3] E. Normand, "Single Event Upset at Ground Level," IEEE Transactions on Nuclear Science, vol. 43, 1996, pp. 2742-2750.

[4] P. Kubalik, R. Dobias and H. Kubatova, "Dependable Design for FPGA based on Duplex System and Reconfiguration", In Proc. of 9th Euromicro Conference on Digital System Design, Los Alamitos: IEEE Computer Society, 2006, pp. 139-145.

[5] D.K. Pradhan, "Fault-Tolerant Computer System Design", Prentice-Hall, Inc., New Jersey, 1996.

[6] P. Fišer, J. Hlavička and H. Kubátová. FC-Min: A Fast Multi-Output Boolean Minimizer, Proc. 29th Euromicro Symposium on Digital Systems Design (DSD'03), Antalya (TR), 1.-6.9.2003, pp. 451-454.

[7] P. Fišer and H. Kubátová, "Output Grouping-Based Decomposition of Logic Functions":, Proc. 8th IEEE Design and Diagnostics of Electronic Circuits and Systems Workshop 2005 (DDECS'05), Sopron, HU, 13.-16.4.2005, pp. 137-144.

[8] P. Kubalík, P. Fišer and H. Kubátová, "Fault Tolerant System Design Method Based on Self-Checking Circuits", Proc. 12th International On-Line Testing Symposium 2006 (IOLTS'06), Lake of Como, Italy, July 10-12, 2006.

[9] E.M. Sentovich et al. "SIS: A System for Sequential Circuit Synthesis", Electronics Research Laboratory Memorandum No. UCB/ERL M92/41, University of California, Berkeley, CA 94720, 1992.

[10] R.K. Brayton, et al. „Logic Minimization Algorithms for VLSI Synthesis", Boston, MA, Kluwer Academic Publishers, 1984.

[11] S. Yang, "Logic Synthesis and Optimization Benchmarks User Guide", Technical Report 1991-IWLS-UG-Saeyang, MCNC, Research Triangle Park, NC, January 1991

[12] J. Cong and K. Minkovich, "Optimality Study of Logic Synthesis for LUT-Based FPGAs", Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on CAD, Vol. 26, Issue 2, Feb. 2007, pp. 230 – 239.

Table 3: Improved availability parameters

Bench	*Single parity*			*Multiple parity*			*Impr. ASS*
	AO	*FS*	*ASS*	*AO*	*FS*	*ASS*	
alu1	3337.5%	100.0%	1	275.00%	100.00%	1	0.0%
apla	40.5%	74.3%	0.999988965	76.19%	87.21%	0.999991356	18.2%
b10	26.7%	92.6%	0,999997416	44.40%	95.83%	0,999998095	3.4%
dk17	41.9%	84.9%	0.999993387	100.00%	95.23%	0.999995824	13.9%
f51m	50.0%	87.2%	0.999993736	72.22%	88.48%	0.999992583	-7.4%
newapla	43.8%	85.3%	0.999993388	75.00%	92.81%	0.999995204	10.7%
newbyte	11.1%	100.0%	1	33.33%	100.00%	1	0.0%
p82	14.7%	85.3%	0.999995793	20.59%	90.33%	0.999996931	6.1%
sex	57.9%	83.6%	0.999991106	84.21%	92.35%	0.999994391	20.4%