POLITECNICO DI TORINO

Dipartimento di Elettronica e Telecommunicazioni

Corso di Laurea in Ingegneria Elettronica

Tesi di Laurea Magistrale

Boh

Relatore
Luciano Lavagno

Daniel Rodas Bautista
matricola: s219976

Luglio 2016

Contents

List of Figures

List of Tables

Acknowledgments

Chapter 1

Introduction

EDA for Reliability and low power. Different methodologies are being analysed to improve the reliability and power consumption of a circuit: -Use of error correcting techniques, from communication theory, in digital design. -Decomposition of the circuit into Reed-Muller form. -Decomposition into cones, increasing redundancy and minimising error propagation throughout the circuit.

Chapter 2

Motivations

2.1 AND-XOR network synthesis

Logic synthesis for low power and more recently reliability is a widely studied subject with many tools being developed throughout industry and academia. State of the art synthesis flows aim at optimizing performance or area or power of a given logic function. During the synthesis process, the redundancy which is inherently captured in its truth table is reduced or indeed removed hence impacting the reliability of the circuit. One can conclude that the most reliable function is represented by the sum of constituent minterms (or sum of products SOP form). However, this un-optimized function leads to a very large area/delay/power overhead given its exponential number of minterms. Fault tolerant techniques for improving reliability of digital circuitry have been of interest from long time. Von Neumann [14] firstly introduced a classification of the error types and proposed some solutions as early as 1956. Important work to cross the field of circuit design with the knowledge of error correction theory has been done by Taylor [13, 12] that used Low Density Parity Check (LDPC) codes to build fault tolerant architectures for reliable systems.

From a gate level perspective, there are two methodologies to improve the reliability of a given circuit while maintaining a minimal overhead. The first method as already presented in [4], is reliability optimization using combinatorial methods and graph manipulations. It was shown that these lead to relatively modest reliability improvements, but with minimal area/delay/power overhead [4]. While the reliability is improved, the circuit is still not fault-tolerant. In this methodology, the number of outputs (or function associated truth table size) is not increased. Instead, the graph is transformed through a sequence of application of a number of rules in order to increase its reliability. A second method of adding redundancy is by modifying the size of the function associated truth table.

In [8, 3], a new methodology called Code Prediction Encoding (CPE) was introduced with the view of extending the logic network with a parity network, hence

increasing the number of outputs of the circuit. The focus of this approach is not on changing the combinational logic but on augmenting it. It enables the retrieval of the correct output even if errors have occurred. The process carries remembrance with the process of adding redundancy to the transmitted message in a communication system. It exploits the afferent topology of the original circuit, which is expanded by embedding an Error Correcting Code (ECC) into its logical functionality.

Any code could be used, hence the generic nature of the technique. This method was successfully applied to improve the reliability of XOR-only logic networks (or linear circuits in our concept) in the context of LDPC encoding [2]. However, the majority of circuits in practice are non-linear in nature as they are built with arbitrary logic gates and hence they are less amenable for a CPE technique as remarked in [8]. Following the rationale of CPE approach, this paper summarizes relevant knowledge regarding the separation of a combinatorial circuit into a linear and a non-linear part. The linear part can be then used in context of CPE. The application of recently published ideas of the vectorial bi-decomposition [10] leads to a partition of circuit parts that contains either only non-linear AND-gates or only linear EXOR-gates. This is particularly interesting as it is known that AND-XOR network results in much better realization and requires fewer product terms than more classical AND-OR realization.

We show that using our decomposition methods, one can derive the Reed-Muller form of a circuit which is known as a hard problem. [5] Another contribution of this work is the quantification of the efficiency of linearization by introducing the notion of degree of linearization. The contributions are outlined on adders of varying size.

Adders are some of the most used non-linear logic functions in practice and numerous architectures were proposed to implement them. We show that even in the case of the symmetric carry function, for which no strong bi-decomposition exists [1, 7], a decomposition into a linear output-part and a non-linear input part could be designed.

We apply the described decomposition at bit level, block level and unit level on an adder architecture and analyze the results in terms of area, delay, power on two technologies, namely ASIC and FPGA.

2.2 CPE

Chapter 3

Synthesis for Reliability by Bi-Decomposition

3.1 Synthesis for Reliability

The application of the CPE approach requires that the circuit structure is split into a linear and a non-linear part. Therefore we explore synthesis methods that find such a separated structure. Figure 3.1 shows the needed architecture of the circuit.

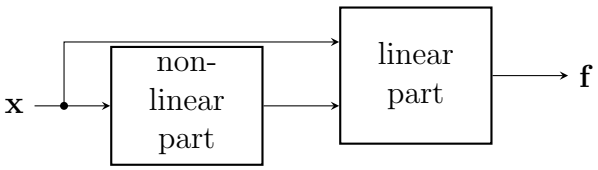

Figure 3.1. Preferred architecture for circuits with a high reliability.

3.1.1 Linearity with Regard to a Variable

The Linearity is a property of a Boolean function which is defined regarding each variable.

Definition 1 (Linearity) *A Boolean function $f(x_i, \mathbf{x}_0)$ is linear with regard to the variable x_i if and only if*

$$\frac{\partial f(x_i, \mathbf{x}_0)}{\partial x_i} = 1 \ . \tag{3.1}$$

If the Boolean function $f(x_i, \mathbf{x}_0)$ satisfies (3.1) this function can be expressed by

$$f(x_i, \mathbf{x}_0) = x_i \oplus g(\mathbf{x}_0) \tag{3.2}$$

The function $g(\mathbf{x}_0)$ can be calculated by

$$g(\mathbf{x}_0) = \max_{x_i} \left(x_i \oplus f(x_i, \mathbf{x}_0) \right) \ , \tag{3.3}$$

using the derivative operation *single maximum* of the BDC [11]. The function $g(\mathbf{x}_0)$ is independent of the variable x_i. The independence of a variable x_i is also a property of a Boolean function.

Definition 2 (Independence) *A Boolean function $f(x_i, \mathbf{x}_0)$ is independent of the variable x_i if and only if*

$$\frac{\partial f(x_i, \mathbf{x}_0)}{\partial x_i} = 0 \ . \tag{3.4}$$

Example 1 (Linear separation of x_i) *The sum function $f_{s_i}(x_i, y_i, c_{i-1})$ of the adder cell i is defined by*

$$f_{s_i}(x_i, y_i, c_{i-1}) = \overline{x}_i \, \overline{y}_i \, c_{i-1} \vee \overline{x}_i \, y_i \, \overline{c}_{i-1} \vee x_i \, \overline{y}_i \, \overline{c}_{i-1} \vee x_i \, y_i \, c_{i-1} \ . \tag{3.5}$$

This function satisfies the condition (3.2) *of the linearity with regard to the variable x_i*

$$\frac{\partial f_{s_i}(x_i, y_i, c_{i-1})}{\partial x_i} = 1 \ , \tag{3.6}$$

which follows form the definition of the derivative

$$\frac{\partial f(x_i, \mathbf{x}_0)}{\partial x_i} = f(x_i = 0, \mathbf{x}_0) \oplus f(x_i = 1, \mathbf{x}_0) \tag{3.7}$$

applied to (3.5)*:*

$$\begin{aligned} \frac{\partial f_{s_i}(x_i, y_i, c_{i-1})}{\partial x_i} &= (\overline{y}_i \, c_{i-1} \vee y_i \, \overline{c}_{i-1} \vee \overline{y}_i \, \overline{c}_{i-1}) \oplus (y_i \, c_{i-1}) \\ &= \overline{y}_i \oplus y_i \\ &= 1 \ . \end{aligned}$$

Using the definition of the maximum with regard to x_i

$$\max_{x_i} f(x_i, \mathbf{x}_0) = f(x_i = 0, \mathbf{x}_0) \vee f(x_i = 1, \mathbf{x}_0) \tag{3.8}$$

the function $g(y_i, c_{i-1})$ and can be calculated based on (3.3)*:*

$$\begin{aligned} g_{s_i}(y_i, c_{i-1}) &= \max_{x_i} \left(x_i \oplus f_{s_i}(x_i, y_i, c_{i-1}) \right) \\ &= (0 \oplus (\overline{y}_i \, c_{i-1} \vee y_i \, \overline{c}_{i-1})) \vee (1 \oplus (\overline{y}_i \, \overline{c}_{i-1} \vee y_i \, c_{i-1})) \\ &= \overline{y}_i \, c_{i-1} \vee y_i \, \overline{c}_{i-1} \ . \end{aligned}$$

Figure 3.2 (a) shows the circuit structure created by this linear separation of the variable x_i.

The function $g(y_i, c_{i-1})$ is linear with regard to both the variables c_{i-i} and y_i. Hence,

$$g(y_i, c_{i-1}) = y_i \oplus c_{i-1}$$

and the function $f_{s_i}(x_i, y_i, c_{i-1})$ is completely linear

$$g(y_i, c_{i-1}) = x_i \oplus y_i \oplus c_{i-1} \ .$$

Figure 3.2 (b) shows the circuit structure created by two linear separations of a single variable using two EXOR-gates.

3.1.2 Degree of Linearity

The derivative with regard to the variables x_i is a Boolean function for which the extreme values specify either the linearity by the value 1 or the independence by the value 0.

The single derivative itself is independent of x_i:

$$\frac{\partial}{\partial x_i}\left(\frac{\partial f(x_i, \mathbf{x}_0)}{\partial x_i}\right) = 0 \ . \tag{3.9}$$

The number of values 1 of the derivative with regard to the variable x_i is a measure for the degree of the linearity.

Definition 3 (Degree of Linearity) *A Boolean function $f(\mathbf{x})$ with n variables $\mathbf{x} = (x_i, \mathbf{x}_0)$ has a* degree of linearity *with regard to the variable x_i in the range $0, \ldots, 1$ defined by*

$$\mathbf{degree}_{x_i}^{lin} f(x_i, \mathbf{x}_0) = \frac{1}{2^{n-1}} * \rho\left(\frac{\partial f(x_i, \mathbf{x}_0)}{\partial x_i}\right) \ , \tag{3.10}$$

where ρ is the number of values 1 of the evaluated function.

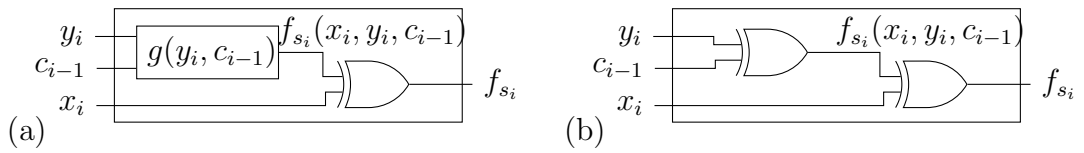

Figure 3.2. Separation of a variable using EXOR gates: (a) with regard to x_i, (b) consecutively with regard to x_i and c_{i-1}

Each variable x_i can be separated from a function $f(x_i, \mathbf{x}_0)$ using an EXOR-gate:

$$f(x_i, \mathbf{x}_0) = x_i \oplus g(x_i, \mathbf{x}_0) \ . \tag{3.11}$$

Each such linear separation satisfies the rule:

$$\mathbf{degree}^{lin}_{x_i} f(x_i, \mathbf{x}_0) = 1 - \mathbf{degree}^{lin}_{x_i} g(x_i, \mathbf{x}_0) \ . \tag{3.12}$$

Hence, the linear separation of a variable x_i can be also useful in the case that $\mathbf{degree}^{lin}_{x_i} f(x_i, \mathbf{x}_0)$ is closed to the value 1 because in this case is the degree of linearity of the function $g(x_i, \mathbf{x}_0)$ much smaller in comparison to $f(x_i, \mathbf{x}_0)$.

Table 3.1. Degree of linearity of adder functions $s_0, \ldots, s_7$

inputs	output functions								
	s_0	s_1	s_2	s_3	s_4	s_5	s_6	s_7	s_8
a_0	1.0000	0.5000	0.2500	0.1250	0.0625	0.0313	0.0156	0.0078	0.0039
b_0	1.0000	0.5000	0.2500	0.1250	0.0625	0.0313	0.0156	0.0078	0.0039
a_1			0.5000	0.2500	0.1250	0.0625	0.0313	0.0156	0.0078
b_1			0.5000	0.2500	0.1250	0.0625	0.0313	0.0156	0.0078
a_2				0.5000	0.2500	0.1250	0.0625	0.0313	0.0156
b_2				0.5000	0.2500	0.1250	0.0625	0.0313	0.0156
a_3					0.5000	0.2500	0.1250	0.0625	0.0313
b_3					0.5000	0.2500	0.1250	0.0625	0.0313
a_4						0.5000	0.2500	0.1250	0.0625
b_4						0.5000	0.2500	0.1250	0.0625
a_5							0.5000	0.2500	0.1250
b_5							0.5000	0.2500	0.1250
a_6								0.5000	0.2500
b_6								0.5000	0.2500
a_7									0.5000
b_7									0.5000

We calculated the degree of linearity for all output functions of an eight bit adder. Table 3.1 enumerates the results. It can be seen that the degree of linearity exponentially decreases with the distance between the index of the output and the index of the input. The low degree of linearity of the low-order increases the needed number of gates in the non-linear part for higher-order sum functions.

3.1.3 Bi-Decompositions

From Table 3.1 can be concluded that the linear separation of a variable is only possible for the function s_0. Alternatively, the wanted XOR-gates for the linear part

can be found by means of the strong XOR-bi-decomposition (see Figure 3.3 (c) on the left). The decomposition functions $g(\mathbf{x}_a, \mathbf{x}_c)$ and $h(\mathbf{x}_b, \mathbf{x}_c)$ are simpler than the given function $f(\mathbf{x}_a, \mathbf{x}_b, \mathbf{x}_c)$ because the function g does not depend on the variables $\mathbf{x}_b$ and the function g does not depend on the variables $\mathbf{x}_a$, respectively. For recursive bi-decomposition in the non-linear also the strong OR-bi-decomposition and the strong AND-bi-decomposition can be used (see Figure 3.3 (a) and (b) on the left).

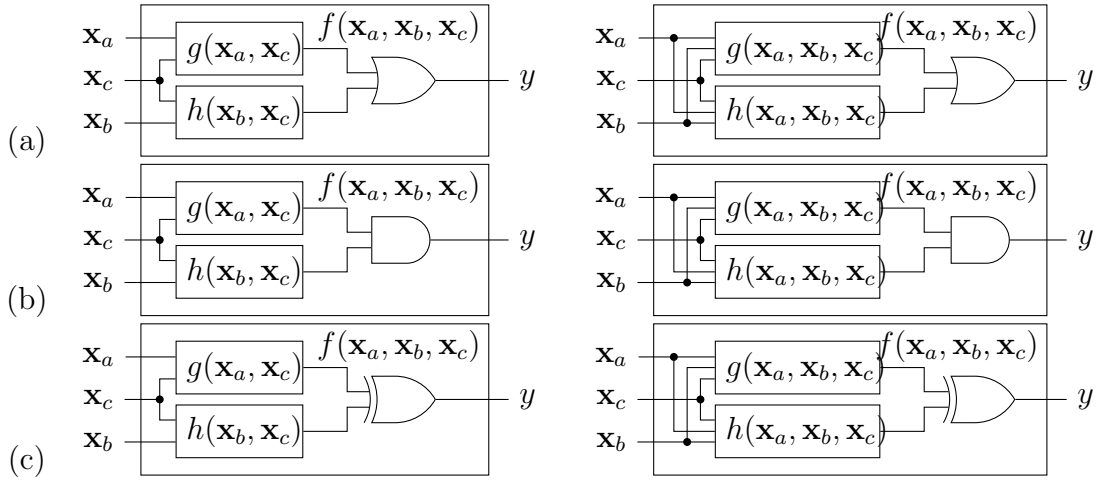

Figure 3.3. Strong bi-decompositions on the left in comparison to the vectorial bi-decomposition on the right using: (a) an OR-gate; (b) an AND-gate; (c) an XOR-gate.

Unfortunately, there are functions for which no strong bi-decomposition exists. The completeness of the bi-decomposition provide the weak OR-bi-decomposition and the weak AND-bi-decomposition [6, 9]. A weak XOR-bi-decomposition exists for each function, but their decomposition function can be even more complex than the given function.

A vectorial bi-decomposition can be exist even if there is no strong bi-decomposition for a given function. Vectorial bi-decompositions were suggested in [10] for the first time. As can be seen in Figure 3.3 on the right, vectorial bi-decompositions exist for an OR-gate, and AND-gate and also for an XOR-gate. The simplifications of the decomposition functions follow from the independence of the simultaneous change of $\mathbf{x}_b$ (3.14) and $\mathbf{x}_a$ (3.15) required in Definition 4.

Definition 4 (Vectorial Bi-Decomposition)
A Boolean function $f(\mathbf{x}_a, \mathbf{x}_b, \mathbf{x}_c)$ is OR-, AND-, or XOR-bi-decomposable with regard to the subsets of variables $\mathbf{x}_a$ and $\mathbf{x}_b$ if

1. *$f(\mathbf{x})$ can be expressed by*

$$f(\mathbf{x}_a, \mathbf{x}_b, \mathbf{x}_c) = g(\mathbf{x}_a, \mathbf{x}_b, \mathbf{x}_c) \bullet h(\mathbf{x}_a, \mathbf{x}_b, \mathbf{x}_c) \tag{3.13}$$

where $\bullet \in \{\vee, \wedge, \oplus\}$,

2. *the decomposition functions* $g(\mathbf{x}_a, \mathbf{x}_b, \mathbf{x}_c)$ *and* $h(\mathbf{x}_a, \mathbf{x}_b, \mathbf{x}_c)$ *satisfy*

$$\frac{\partial g(\mathbf{x}_a, \mathbf{x}_b, \mathbf{x}_c)}{\partial \mathbf{x}_b} = 0 \ , \tag{3.14}$$

$$\frac{\partial h(\mathbf{x}_a, \mathbf{x}_b, \mathbf{x}_c)}{\partial \mathbf{x}_a} = 0 \ . \tag{3.15}$$

Example 2 (Vectorial bi-decomposition of a symmetric function)
The carry function $f_{c_i}(x_i, y_i, c_{i-1})$ *of an adder is a symmetric function. It is known that there is no strong bi-decomposition for symmetric function. Figure 3.4 shows the result of a vectorial XOR-bi-decomposition of this function. For one of the decomposition functions even exists a strong XOR-bi-decomposition, sot that the linear part of the consists of two XOR-gates.*

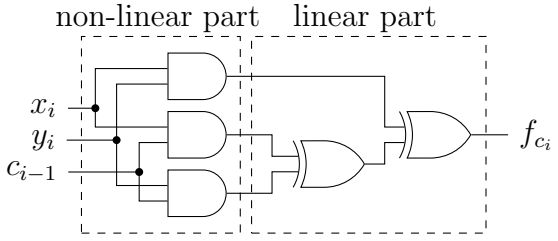

Figure 3.4. Carry function $f_{c_i}(x_i, y_i, c_{i-1})$ decomposed by a vectorial EXOR-bi-decomposition followed by strong EXOR-bi-decomposition.

Table 3.2. Gate equivalents for the decomposed adder

	Architecture on Figure 3.1			Standard RCA
Number of Bits	Linear Part	Nonlinear Part	Total	Total
4	32	17	49	34
5	65	39	105	43
6	132	86	218	53
7	266	180	446	62
8	534	371	904	72
10	2,138	1,520	3,658	91
11	4,278	3,054	7,332	100
16	136,923	98,279	235,202	148

As it can be remarked from this table, the gate size is doubled with every bit increase. This is due to the process of internal carry generation which grows exponentially.

Table 3.3. Power for the decomposed adder

Number of Bits	Internal Power	Switching Power	Leakage Power	Total Power
4	6.5159e-03 mW	8.4444e-04 mW	392.5974 nW	7.7530e-03 mW
5	1.0973e-02 mW	1.8577e-03 mW	822.0242 nW	1.3653e-02 mW
6	1.8275e-02 mW	3.3673e-03 mW	1.6732e+03 nW	2.3315e-02 mW
7	3.2342e-02 mW	6.2124e-03 mW	3.4077e+03 nW	4.1962e-02 mW
8	5.6895e-02 mW	1.1168e-02 mW	6.8369e+03 nW	7.4900e-02 mW
10	0.1868 mW	3.4323e-02 mW	2.7489e+04 nW	0.2486 mW
11	0.3493 mW	6.1830e-02 mW	5.5015e+04 nW	0.4662 mW
16	10.0527 mW	1.6090 mW	1.7603e+06 nW	13.4228 mW

Table 3.4. Areas and maximal time delay for the two implementations of the decomposed adders

bits	Areas (in μm)		Time Delay (in ns)	
	Global	Local	Global	Local
4	70	49	0.62	0.70
5	150	68	1.00	0.97
6	314	87	1.69	1.24
7	643	105	3.24	1.50
8	1,302	124	5.97	1.77
9	2,623	143	11.38	2.03
10	5,268	162	22.25	2.30
11	10,558	180	43.75	2.56
16	338,690	274	1379.43	3.89

3.2 Experimental Results - ASIC and FPGA Synthesis

Due the results of Subsection 3.1.1 we compare two synthesis approaches. The first one uses the architecture shown in Figure 3.1 for complete adders of different numbers of bits (similar to a carry look-ahead adder (CLA)). The second one uses vectorial and strong bi-decompositions as shown in Figure 3.4 for adders with a restricted number of bits and generates larger adders in an architecture similar to a ripple carry adder (RCA). In both architectures we assume that there is no carry in and carry out.

3.2.1 ASIC Synthesis

The adders were synthesized in 65nm CMOS technology using the Design Compiler from Synopsys. The global decomposition results in very large area of implementation as noted in Table 3.2. Table 3.3 shows the power associated to implementations on ASIC of the globally decomposed adder for different number of bits.

Table 3.4 compares both the area and the maximum delay for the two different implementations of the decomposed adder. We can see that the local decomposition results in much smaller area. Moreover, the maximum delay in the local decomposition is significantly shorter. For these reasons the FPGA synthesis was then done only for the locally decomposed adder.

3.2.2 FPGA Synthesis

Tables 3.5, 3.6 and 3.7 show the area, power, and timing associated to implementations on FPGA of the decomposed adder for different number of bits and compares them to the equivalent values for a standard adder. Synthesis and routing was done for a Zynq FPGA using the Vivado design suite.

Table 3.5. Number of LUTs used

bits	decomposed	undecomposed	bits	decomposed	undecomposed
4	4	4	19	20	19
5	4	4	20	21	20
6	6	6	21	22	21
7	6	7	22	23	22
8	8	8	23	24	23
9	8	9	24	25	24
10	10	10	25	26	25
11	10	11	26	27	26
16	42	16	31	32	31

While for the locally decomposed adder the number of gates and associated delay is drastically improved, one needs to consider further optimization for the circuits such that the presented architectures become competitive. For the locally decomposed adder a particular emphasis will be on implementations on FPGA and power optimization. Further study into using academic tools for logic synthesis and optimization such as Expresso, SIS, ABC, etc. will be also performed.

Table 3.6. Power

bits	decomposed	standard	bits	decomposed	standard
4	2.399	2.399	19	13.283	14.187
5	3.068	3.068	20	13.936	14.922
6	3.741	4.054	21	14.610	15.655
7	4.432	4.813	22	15.272	16.388
8	5.134	5.591	23	15.942	17.119
9	5.847	6.395	24	16.631	17.857
10	6.583	7.226	25	17.285	18.564
11	7.347	8.105	26	17.745	19.202
12	8.153	8.974	27	18.427	19.951
13	8.945	9.702	28	19.081	20.701
14	12.222	10.441	29	19.734	21.465
15	13.159	11.172	30	20.412	22.212
16	14.021	11.909	31	21.104	22.957
17	15.241	12.739	32	21.784	23.727
18	16.092	13.470			

Table 3.7. Timing (max delay path in ns)

bits	decomposed	standard	bits	decomposed	standard
4	7.875	7.875	19	26.894	10.306
5	8.321	8.321	20	26.327	10.052
6	8.278	8.348	21	28.798	10.252
7	8.729	8.371	22	30.979	10.518
8	9.074	8.689	23	31.079	10.506
9	10.462	8.518	24	31.556	10.52
10	10.707	8.958	25	35.130	10.212
11	12.007	8.956	26	34.476	10.634
12	12.258	9.082	27	34.620	10.628
13	13.345	8.903	28	36.111	10.865
14	18.462	9.162	29	33.641	10.739
15	19.567	9.427	30	38.501	10.847
16	19.486	9.578	31	39.772	11.074
17	21.299	9.911	32	40.908	11.203
18	22.309	10.052			

Chapter 4

CPE

Chapter 5

Integration

Chapter 6

Results

Chapter 7

Conclusions

With the continuing logic technology scaling, design for reliability/power/test is becoming a major concern. In this context, a significant effort across recent years was in efficient decomposition of logic functions in AND/XOR or OR/XOR networks. The strong EXOR-bi-decomposition is an alternative method to split a give Boolean function into simpler decomposition function using an EXOR-gate. The vectorial bi-decomposition extents the possibilities for decomposition. We show that even in the case of the symmetric carry function, for which no strong bi-decomposition exists, a decomposition into a linear output-part and a non-linear input part could be designed. A Reed-Muller form for the adder can be achieved. Also, some quantification in terms of degree of linearization is given. Using the proposed decomposition, we implement the resulting decomposed adders on ASIC and FPGA technologies and give some preliminary results. The results show that further development of custom optimization tools is required. This is part of future work together with the integration of code prediction encoding for improved reliability.

Bibliography

[1] Dieter Bochmann, Frank Dresig, and Bernd Steinbach. A new decomposition method for multilevel circuit design. In *Proceedings of the Conference on European Design Automation*, EDAC '91, pages 374–377. IEEE Computer Society, 1991.

[2] E. Dupraz, V. Savin, S. K. Grandhi, E. Popovici, and D. Declerq. Practical ldpc encoders robust to hardware errors. *accepted IEEE ICC*, 2016.

[3] S. Grandhi, E. Dupraz, C. Spagnol, V. Savin, and E. Popovici. Cpe: Codeword prediction encoder. *accepted IEEE European Test Symposium*, May 2016.

[4] S. K. Grandhi, D. McCarthy, C. Spagnol, E. Popovici, and S. Cotofana. Rost-c: Reliability driven optimisation and synthesis techniques for combinational circuits. *Computer Design (ICCD), 2015 33rd IEEE International Conference on*, pages 431–434, 2015.

[5] D. Knysh and E. Dubrova. Rule-based optimisation of and-xor expressions. *Facta Universitatis*, 24(3):437–449, 2011.

[6] Trung Quoc Le. *Testbarkeit kombinatorischer Schaltungen - Theorie und Entwurf*. PhD thesis, Technical University of Karl-Marx-Stadt, Germany, 1989. written in German, English title: Testability of Combinational Circuits - Theory and Design.

[7] Alan Mishchenko, Bernd Steinbach, and Marek Perkowski. An algorithm for bi-decomposition of logic functions. In *Proceedings of the 38th Annual Design Automation Conference*, DAC '01, pages 103–108, New York, NY, USA, 2001. ACM.

[8] Emanuel Popovici, Satish Grandhi, Christian Spagnol, Valentin Savin, and Sorin Cotofana. Fault tolerant synthesis through error correcting codes driven graph augmentationg. *FP7-ICT/FET-OPEN/ i-RISC project, Deliverable 5.2*, February 2015.

[9] Christian Posthoff and Bernd Steinbach. *Logic Functions and Equations - Binary Models for Computer Science*. Springer, Dordrecht, The Netherlands, 2004.

[10] Bernd Steinbach. Vectorial bi-decompositions of logic functions. In *Proceedings of the Reed-Muller Workshop 2015*, number 4 in RM '15, pages 1–10, 2015.

[11] Bernd Steinbach and Christian Posthoff. Boolean differential calculus. In Tsutomu Sasao and Jon T. Butler, editors, *Progress in Applications of Boolean Functions*, pages 55–78. Morgan & Claypool Publishers, San Rafael, CA, USA, 2010.

[12] Michael Taylor. Reliable computation in computing systems designed from unreliable components. *Bell Syst. Tech. J.*, 47:2339–2266, December 1968.

[13] Michael Taylor. Reliable information storage in memories designed from unreliable components. *Bell Syst. Tech. J.*, 47:2299–2337, 1968.

[14] John Von Neumann. Probabilistic logics and the synthesis of reliable organisms from unreliable components. *Automata Studies*, 34:43–98, 1956.