TESI DI DOTTORATO

UNIVERSITÀ DEGLI STUDI DI NAPOLI "FEDERICO II"

DIPARTIMENTO DI INGEGNERIA BIOMEDICA,
ELETTRONICA E DELLE TELECOMUNICAZIONI

DOTTORATO DI RICERCA IN
INGEGNERIA ELETTRONICA E DELLE TELECOMUNICAZIONI

TRUNCATED BINARY MULTIPLIERS WITH MINIMUM MEAN SQUARE ERROR: ANALYTICAL CHARACTERIZATION, CIRCUIT IMPLEMENTATION AND APPLICATIONS

VALERIA GAROFALO

Il Coordinatore del Corso di Dottorato	Il Tutore
Ch.mo Prof. Niccolò RINALDI	Ch.mo Prof. Ettore NAPOLI

A. A. 2008–2009

To Fabrizio and
to my grandparents

Acknowledgments

First of all I would like to thank my tutor, Professor Ettore Napoli, for his constant support, his helpful suggestions, his teaching that has been crucial to achieve this goal.
Many thanks to Davide De Caro, who has always been helpful and able to solve all my problems.
A special thanks to Nicola Petra, for stimulating my activity and my interest with suggestions and discussions, for his friendship and support. The time spent in conferences wouldn't have been so great without him!
Many thanks to Professor Antonio Strollo for his priceless lessons.
I also want to thank Professor Florin Udrea for decisive help during the time spent in Cambridge and very useful meetings.
Without these people my experience through the years of the Ph.D wouldn't have been possible at all.

Thanks to HVM group of the University of Cambridge, Sumita, Prasanta, Marina, Maryline, Hatice, Wesley, Zeeshan, Floran, and to Serena and Dominik.

A really big thanks goes to the dear friends of the DIBET, Ilaria, Grazia, Michele, Maurizio, Pierluigi, Marino, Enzo, Salvatore, Matteo, Lucio, Dino, for the pleasant and funny moments we spent together, but most of all for their friendship.

I would like to thank all the people who support me since always, with great patience, great love... Alessio, Marco, Sara, Daniela, Gianfranco, Federica, Valentina, Marianne.

The final most important thanks goes to my parents and my sister. They make everything possible, they are my strength, my joy.

Contents

List of Figures

List of Tables

Notations

X	Multiplicand $(x_1 x_2 \dots x_{n-1} x_n)$
Y	Multiplier $(y_1 y_2 \dots y_{n-1} y_n)$
P	Product Full-width Multiplier $(p_1 p_2 \dots p_{2n-1} p_{2n})$
P_t	Product Truncated Multiplier $(p_{t,1} p_{t,2} \dots p_{t,n-1} p_{t,n})$
$\bigvee$	Boolean operator *OR*
$\bigwedge$	Boolean operator *AND*
$\oplus$	Boolean operator *XOR*
MSP	n-1 most significant columns of the Partial Products matrix
LSP	n less significant columns of the Partial Products matrix
IC	Input Correction vector ($(n+h+1)^{th}$ column)
γ_i	Generic element of the IC
K_{round}	Rounding constant
$f(\text{IC})$	Compensation function
$\mathbf{lsb}$	Weight of the less-significant bit of the result $[2^{-n}]$

$\mathbf{lsb_{IC}}$	Weight of the IC vector [2^{-n-h-1}]
$\mathbf{lsb_q}$	Weight of the less-significant bit of f(IC) [2^{-n-m}]
$n_{eq} = n - h$	Number of columns in $\text{LSP}_{\text{minor}}$
Θ	Set of all possible values of the vector IC
$\Omega(A)$	Subset of all couples (x, y) which give $IC = A$
$\mu_{\text{LSP}}(A)$	Mean value of $\text{LSP}_{\text{minor}}$ in $\Omega(A)$
$\sigma^2_{\text{LSP}}(A)$	Variance of $\text{LSP}_{\text{minor}}$ in $\Omega(A)$
e_{round}	Rounding error
μ_{round}	Mean value of the rounding error
$\varepsilon^2_{\text{round}}$	Variance of the rounding error
e_{total}	Error of the truncated multiplier [$e_{\text{total}} = P - P_t$]
μ_{total}	Mean value of e_{total}
$\varepsilon^2_{\text{total}}$	Mean square error of e_{total}
σ^2_{total}	Variance of e_{total}
e_{trunc}	Truncation error
μ_{trunc}	Mean value of e_{trunc}
σ^2_{trunc}	Variance of e_{trunc}
e_{erasing}	Error related to f(IC), [$e_{\text{erasing}} = f(\text{IC}) - S_{\text{LSP}_{\text{minor}}}$]
μ_{erasing}	Mean value of e_{erasing}
$\varepsilon^2_{\text{erasing}}$	Mean square error of e_{erasing}

$\sigma^2_{\text{erasing}}$	Variance of e_{erasing}
$\varepsilon^2_{\text{intrinsic}}$	Intrinsic Error, independent by $f(\text{IC})$
e_{comp}	Compensation error $(f(\text{IC}) - \mu_{\text{LSP}}(IC))$
μ_{comp}	Mean value of e_{comp}
$\varepsilon^2_{\text{comp}}$	Mean square value of e_{comp}
σ^2_{comp}	Variance of e_{comp}
$f_{opt}(\text{IC})$	Optimal Compensation Function
$\varepsilon^2_{\text{low\_bound}}$	Lower error bound of truncated multiplier
$f_{lin}(\text{IC})$	Linear Correction Function
μ_{lin}	Mean value of $(f_{lin}(\text{IC}) - \mu_{\text{LSP}}(IC))$
σ^2_{lin}	Variance of $(f_{lin}(\text{IC}) - \mu_{\text{LSP}}(IC))$
$f_q(\text{IC})$	Quantized-Coefficient correction function
$\Delta f_{lin}(\text{IC})$	Difference between $f_q(\text{IC})$ and $f_{lin}(\text{IC})$
μ_{quant}	Mean value of $\Delta f_{lin}(\text{IC})$
σ^2_{q}	Variance of $\Delta f_{lin}(\text{IC})$
ε_{max}	Maximum absolute error
ε_{FIR}	Error of the FIR filter with truncated MAC
μ_{FIR}	Mean error of ε_{FIR}
$\varepsilon^2_{\text{FIR}}$	Mean square error of ε_{FIR}
σ^2_{FIR}	Variance of ε_{FIR}

Introduction

In the wireless multimedia word, DSP systems are ubiquitous. DSP algorithms are computationally intensive and test the limits of battery life in portable device such as cell phones, hearing aids, MP3 players, digital video recorders and so on. Multiplication and squaring are the main operation in many signal processing algorithms (filtering, convolution, FFT, DCT, euclidean distance etc.), hence efficient parallel multipliers are desirable.

A full-width digital $n \times n$ bits multiplier computes the $2n$ bits output as a weighted sum of partial products. A multiplier with the output represented on n bits output is useful, as example, in DSP datapaths which saves the output in the same n bits registers of the input. Note that the truncated multipliers are useful not only for DSP but also for digital, computational intensive, ASICs where the bit-widths at the output of the arithmetic blocks are chosen on the basis of system-related accuracy issues. Hence $2n$ bits of precision at the multiplier output are very often more than required.

A truncated multiplier is an $n \times n$ multiplier with n bits output. Since in a truncated multiplier the n less-significant bits of the full-width product are discarded, some of the partial products are removed and replaced by a suitable compensation function, to trade-off accuracy with hardware cost. Several techniques have been proposed in the Literature following this basic idea. The difference between the various circuits is in the choice and the implementation of the compensation circuit.

The correction techniques proposed in the Literature are obtained through exhaustive search. This means that the results are only available for small n values and that the proposed approach are not extendable to greater bit widths. Furthermore the analytical characterization of the error is not possible.

In this dissertation an innovative solution for the design and characterization of truncated multipliers is presented.

The proposed circuits are based on the analytical calculation of the error of the truncated multiplier. This approach allows to have the description of a mul-

tiplier characterized by a minimum mean square error which gives a fast and low power VLSI implementation. Furthermore the analytical approach yields to a closed form expression of the mean square error and maximum absolute error for the proposed truncated multipliers. In this way the a priori knowledge of the output error is available. The errors are known for every bit width of the multiplier and it is also possible to decide, for a given bit width, which correction circuit has to be used in order to obtain a certain error. This analytical relation between the error and the parameters of hardware implementation is extremely important for the digital designer, since now it is possible to select the suitable implementation as a function of the desired accuracy.

Proposed truncated multipliers overcome the previously proposed truncated multipliers since provide lower error, lower power dissipation, lower area occupation and also provide higher working frequency. The circuits are also easily implemented and allow an automatic HDL description as a function of bit width and desired error. The complete description of the errors for the truncated multipliers allows the use of these circuits as building blocks for more complex systems. It will be shown how the proposed multiplier can be used to design low area occupation FIR filters and an efficient PI temperature controller.

Dissertation outline

The outline of the thesis is the following:

Chapter 1 summarizes the state of art full-width and truncated multipliers. The two existent methods for the implementation of the truncated multipliers, constant correction and variable correction, are described and analyzed.

Chapter 2 describes the analytical characterization of the proposed truncated multiplier. First the optimal compensation function $f_{opt}(\mathrm{IC})$, that minimizes the mean square error of the truncated multiplier, is analytically calculated and characterized. Since $f_{opt}(\mathrm{IC})$ is a quadratic form, with the help of the developed theory, the derivation of a sub-optimal compensation function, $f_{lin}(\mathrm{IC})$ (first grade approximation) is described. In the Chapter it is shown that the linear compensation function yields error performances very similar to the one achievable with $f_{opt}(\mathrm{IC})$ and has the additional advantage of an

easy hardware implementation. Finally the aspects related to the quantization of the coefficient of $f_{lin}(\mathrm{IC})$ are investigated. The expression, in closed form, of the intrinsic error, mean square error and maximum absolute error of the multiplier are given.

Chapter 3 deals with the practical implementation of the quantized linear compensation function proposed in the Chapter 2. The performances of the new truncated multipliers are extensively compared with previously proposed circuits in terms of maximum absolute error, mean square error, hardware performances.

Chapter 4 extends the methodology described in Chapter 2 for the truncated multipliers to the truncated squarer. After a brief introduction about the state of the art squarer, the optimal compensation function, its linear approximation and quantized version are shown. Finally the results of the simulations and hardware implementation are presented and a comparison with the existent techniques is highlighted.

Chapter 5 is focused on a typical DSP application, FIR filter. In this chapter the analysis of the impact of the use of truncated multiplier in FIR filter is presented. The parameters of the truncated multipliers that are more significant for the optimization of a digital filter are defined. In order to evaluate the performances of the proposed LMS multiplier in this application, a comparison with the state of art multiplier is carried out in terms of error, power, frequency and area occupation.

Chapter 6 describes another application where is possible to use the truncated multipliers: temperature control. In this chapter the implementation of a PI temperature controller for a gas sensor using the LMS truncated multiplier is described. This implementation is compared with the traditional PI controller, the OnOff controller, the mixed controller. Finally two chips are presented.

Publications

- V. Garofalo, E. Napoli, A.G.M. Strollo, ”Code compression for ARM7 embedded systems”. European Conference on Circuit Theory and De-

sign. 26-30 August 2007

- E. Della Sala, E. Sciagura, D. De Caro, A. Caravella, P. Longobardi, P. Zicari, V. Garofalo, G. Chapuano, P. Corsonello, E. Napoli, A.G.M. Strollo, "High Rate Data Down Link." proc. of 18th ESA Symposium for Sounding Rockets and Balloons, June, 2007

- V. Garofalo, N. Petra, D. De Caro, A.G.M. Strollo, E. Napoli, "Low error Truncated Multipliers for DSP applications", IEEE International Conference on Electronics, Circuits, and Systems 1-3 September 2008

- A.G.M. Strollo, D. De Caro, N. Petra, E. Napoli, V. Garofalo, "Constrained Piecewise Polinomial Approximation for Hardware Implementation of Elementary Functions ", IEEE International Conference on Electronics, Circuits, and Systems 1-3 Sept. 2008

- V. Garofalo, "Fixed-width multipliers for the implementation of efficient digital FIR filters", Microelectronics Journal, vol. 39; p. 1491-1498, ISSN: 0959-8324 (doi:10.1016/j.physletb.2003.10.071)

- N. Petra, D. De Caro, V. Garofalo, E. Napoli, A.G.M. Strollo, "Truncated Binary Multipliers with Optimal Compensation Function", accepted to IEEE Transaction on Circuits and Systems I

- V. Garofalo, P.K. Guha, S.Z. Ali, S. Santra, E. Napoli, F. Udrea "Mixed Signal Temperature Control Circuit for On-Chip CMOS Gas Sensor", International Semiconductor Conference 2009, October 2009, Sinaia, Romania

- S.Z. Ali, S. Santra, P.K. Guha, I. Haneef, V. Garofalo, C. Schwandt, J.A. Covington, R.V. Kumar, J.W. Gardner, W.I. Milne, F. Udrea, "Nanowire Hydrogen Gas Sensor Employing CMOS Micro-Hotplate ", IEEE SENSORS 2009 Conference, 25-28 October 2009, New Zealand

- V. Garofalo, N. Petra, E. Napoli, "Analytical calculation of the maximum error for a family of truncated multipliers providing minimum mean square error", submitted to IEEE Transaction on Computers

- V. Garofalo, M. Coppola, D. De Caro, E. Napoli, N. Petra, A.G.M. Strollo, "A Novel Truncated Squarer with Linear Compensation Function", submitted to IEEE Internat Symposium on Circuits and Systems (ISCAS) 2010

- N. Petra, D. De Caro, A. G. M. Strollo, V. Garofalo, E. Napoli, M. Coppola, P. Todisco, ”Fixed-Width CSD Multipliers With Minimum Mean Square Error”, submitted to IEEE ISCAS 2010
- A.G.M. Strollo, D. De Caro, N. Petra, E. Napoli, M. Coppola, V. Garofalo, ”Non Uniform Piecewise-Linear Approximation For High Performance Direct Digital Frequency Synthesizers”, submitted to IEEE ISCAS 2010
- D. De Caro, M. Coppola, N. Petra, E. Napoli, A.G.M. Strollo, V. Garofalo, ”High Speed Differential Resistor Ladder for A/D Converters”, submitted to IEEE ISCAS 2010

Chapter 1

Binary Multiplication

Multiplication is one of the most area consuming arithmetic operations in high-performance circuits. As a consequence many research works deal with low-power design of high-speed multipliers. Multiplication involves two basic operations, the generation of the partial products and their sum, performed using two kinds of multiplication algorithms, serial and parallel. Serial multiplication algorithms use sequential circuits with feedbacks: inner products are sequentially produced and computed. Parallel multiplication algorithms often use combinational circuits and do not contain feedback structures.

A full-width digital $n \times n$ multiplier computes the $2n$ bits output as a weighted sum of partial products. When an application requires a multiplication output with n bits, it is substituted by a truncated multiplier, which is a $n \times n$ multiplier with n bits output. After a short description of the generation of partial product done in Sec. 1.1, Sec. 1.2 describes the common techniques used in order to implement full-width multipliers. Finally Sec. 1.3 summarizes the state of art for truncated multipliers.

1.1 Partial-Product Generation

The following notation is used in our discussion of multiplication algorithm:

X	Multiplicand	$(x_1 x_2 \dots x_{n-1} x_n)$
Y	Multiplier	$(y_1 y_2 \dots y_{n-1} y_n)$
P	Product Full-width Multiplier $(X \cdot Y)$	$(p_1 p_2 \dots p_{2n-1} p_{2n})$
P_t	Product Truncated Multiplier $(X \cdot Y)_t$	$(p_{t,1} p_{t,2} \dots p_{t,n-1} p_{t,n})$

WEIGHT	2^{-1}	2^{-2}	2^{-3}	2^{-4}	2^{-5}	2^{-6}	2^{-7}	2^{-8}	2^{-9}	2^{-10}	2^{-11}	2^{-12}	2^{-13}	2^{-14}	2^{-15}	2^{-16}
MULTIPLICAND	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8								
MULTIPLIER	y_1	y_2	y_3	y_4	y_5	y_6	y_7	y_8								
PARTIAL PRODUCTS									x_1y_8	x_2y_8	x_3y_8	x_4y_8	x_5y_8	x_6y_8	x_7y_8	x_8y_8
								x_1y_7	x_2y_7	x_3y_7	x_4y_7	x_5y_7	x_6y_7	x_7y_7	x_8y_7	
							x_1y_6	x_2y_6	x_3y_6	x_4y_6	x_5y_6	x_6y_6	x_7y_6	x_8y_6		
						x_1y_5	x_2y_5	x_3y_5	x_4y_5	x_5y_5	x_6y_5	x_7y_5	x_8y_5			
					x_1y_4	x_2y_4	x_3y_4	x_4y_4	x_5y_4	x_6y_4	x_7y_4	x_8y_4				
				x_1y_3	x_2y_3	x_3y_3	x_4y_3	x_5y_3	x_6y_3	x_7y_3	x_8y_3					
			x_1y_2	x_2y_2	x_3y_2	x_4y_2	x_4y_2	x_5y_2	x_6y_2	x_8y_2						
		x_1y_1	x_2y_1	x_3y_1	x_4y_1	x_4y_1	x_5y_1	x_6y_1	x_8y_1							
PRODUCT	p_1	p_2	p_3	p_4	p_5	p_6	p_7	p_8	p_9	p_{10}	p_{11}	p_{12}	p_{13}	p_{14}	p_{15}	p_{16}

Figure 1.1: Unsigned Partial Products matrix, $n = 8$.

Parallel multipliers first generate partial products, then add them together to produce the product. In the following it will be described the generation of Partial Products for unsigned, two's complement, and mixed operand multipliers.

1.1.1 Unsigned Multiplication

Without loss of generality, let's assume that the inputs of the multiplier represent *n* bit unsigned fractional value in $[0, 1)$:

$$X = \sum_{i=1}^{n} x_i \cdot 2^{-i} \tag{1.1}$$

$$Y = \sum_{i=1}^{n} y_i \cdot 2^{-i} \tag{1.2}$$

The output of the multiplier $P = X \cdot Y$ is computed as:

$$P = \sum_{i=1}^{2n} p_i \cdot 2^{-i} = \sum_{i=1}^{n} \sum_{j=1}^{n} x_i y_j \cdot 2^{-i-j} \tag{1.3}$$

Fig. 1.1 shows the matrix of the Partial Products (PPs) $x_i y_j$ for a 8×8 multiplier. Each row of the matrix corresponds to the products of the multiplicand X and a single bit of the multiplier Y. Generation of the partial products for unsigned multiplication can be implemented with n^2 *AND* gates.

WEIGHT	-2^{-1}	2^{-2}	2^{-3}	2^{-4}	2^{-5}	2^{-6}	2^{-7}	2^{-8}	2^{-9}	2^{-10}	2^{-11}	2^{-12}	2^{-13}	2^{-14}	2^{-15}	2^{-16}
MULTIPLICAND	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8								
MULTIPLIER	y_1	y_2	y_3	y_4	y_5	y_6	y_7	y_8								
PARTIAL PRODUCTS								1	$\overline{x_1y_8}$	x_2y_8	x_3y_8	x_4y_8	x_5y_8	x_6y_8	x_7y_8	x_8y_8
								$\overline{x_1y_7}$	x_2y_7	x_3y_7	x_4y_7	x_5y_7	x_6y_7	x_7y_7	x_8y_7	
							$\overline{x_1y_6}$	x_2y_6	x_3y_6	x_4y_6	x_5y_6	x_6y_6	x_7y_6	x_8y_6		
						$\overline{x_1y_5}$	x_2y_5	x_3y_5	x_4y_5	x_5y_5	x_6y_5	x_7y_5	x_8y_5			
					$\overline{x_1y_4}$	x_2y_4	x_3y_4	x_4y_4	x_5y_4	x_6y_4	x_7y_4	x_8y_4				
				$\overline{x_1y_3}$	x_2y_3	x_3y_3	x_4y_3	x_5y_3	x_6y_3	x_7y_3	x_8y_3					
			$\overline{x_1y_2}$	x_2y_2	x_3y_2	x_4y_2	x_5y_2	x_6y_2	x_8y_2							
	1	x_1y_1	$\overline{x_2y_1}$	$\overline{x_3y_1}$	$\overline{x_4y_1}$	$\overline{x_4y_1}$	$\overline{x_5y_1}$	$\overline{x_6y_1}$	$\overline{x_8y_1}$							
PRODUCT	p_1	p_2	p_3	p_4	p_5	p_6	p_7	p_8	p_9	p_{10}	p_{11}	p_{12}	p_{13}	p_{14}	p_{15}	p_{16}

Figure 1.2: Two's complement Partial Products matrix, $n = 8$.

1.1.2 Two's Complement Multiplication

The unsigned multiplication matrix can be modified for operation on two's complement operands using the technique presented by Baugh and Wooley [1]. If the inputs of the multiplier represent *n* bit two's complement fractional value in $[0, 1)$:

$$X = -x_1 \cdot 2^{-1} + \sum_{i=2}^{n} x_i \cdot 2^{-i} \tag{1.4}$$

$$Y = -y_1 \cdot 2^{-1} + \sum_{i=2}^{n} y_i \cdot 2^{-i} \tag{1.5}$$

the output of the full-width multiplier $P = XY$ is computed as:

$$\begin{aligned} P = &- p_1 \cdot 2^{-1} + \sum_{i=2}^{2n} p_i \cdot 2^{-i} = x_1 y_1 2^{-2} + \sum_{i=2}^{n} \sum_{j=2}^{n} x_i y_j \cdot 2^{-i-j} \\ &- \sum_{i=2}^{n} x_1 y_i \cdot 2^{-i-1} - \sum_{i=2}^{n} x_i y_1 \cdot 2^{-i-1} \end{aligned} \tag{1.6}$$

The first two terms of eq. (1.6) are positive, while the last two terms are either zero or negative. In order to calculate the product, instead of subtract the last two terms it is possible to add the opposite values. Since the representation is in two's complement, the opposite is easily calculated considering all the bit

complemented and adding 1 in the less significant column:

$$-\sum_{i=2}^{n} x_1 y_i \cdot 2^{-i-1} = \sum_{i=2}^{n} (\overline{x_1 y_i}) \cdot 2^{-i-1} + 2^{-2} + 2^{-n-1} \tag{1.7}$$

$$-\sum_{i=2}^{n} x_i y_1 \cdot 2^{-i-1} = \sum_{i=2}^{n} (\overline{x_i y_1}) \cdot 2^{-i-1} + 2^{-2} + 2^{-n-1} \tag{1.8}$$

note that in eq. (1.7)-(1.8) it is present also the 1 (whose weight is 2^{-2}) due to the sign extension. Finally one obtains:

$$P = 2^{-1} + 2^{-n} + x_1 y_1 2^{-2} + \sum_{i=2}^{n} \sum_{j=2}^{n} x_i y_j \cdot 2^{-i-j} + \sum_{i=2}^{n} (\overline{x_i y_1} + \overline{x_1 y_i}) \cdot 2^{-i-1} \tag{1.9}$$

Fig. 1.2 shows the matrix of the Partial Products (PPs) $x_i y_j$ for a 8×8 multiplier. Generation of the partial products for two's complement multiplication can be implemented with $\left((n-1)^2 + 1\right)$ *AND* gates and $(2n-2)$ *NAND* gates.

1.1.3 Mixed-Operand Multiplication

If X is a n bit unsigned fractional number and Y is n bit two's complement fractional number:

$$X = \sum_{i=1}^{n} x_i \cdot 2^{-i} \tag{1.10}$$

$$Y = -y_1 \cdot 2^{-1} + \sum_{i=2}^{n} y_i \cdot 2^{-i} \tag{1.11}$$

following the same reasoning of Sec. 1.1.2, the final product can be computed as:

$$P = 2^{-1} + 2^{-n-1} + \sum_{i=1}^{n} \sum_{j=2}^{n} x_i y_j \cdot 2^{-i-j} + \sum_{i=1}^{n} \overline{x_i y_1} \cdot 2^{-i-1} \tag{1.12}$$

Fig. 1.3 shows the matrix of the Partial Products (PPs) $x_i y_j$ for a 8×8 multiplier. Generation of the partial products for mixed-operand multiplication can be implemented with $n(n-1)$ *AND* gates and n *NAND* gates.

WEIGHT	-2^{-1}	2^{-2}	2^{-3}	2^{-4}	2^{-5}	2^{-6}	2^{-7}	2^{-8}	2^{-9}	2^{-10}	2^{-11}	2^{-12}	2^{-13}	2^{-14}	2^{-15}	2^{-16}
MULTIPLICAND	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8								
MULTIPLIER	y_1	y_2	y_3	y_4	y_5	y_6	y_7	y_8								
PARTIAL PRODUCTS									x_1y_8	x_2y_8	x_3y_8	x_4y_8	x_5y_8	x_6y_8	x_7y_8	x_8y_8
								x_1y_7	x_2y_7	x_3y_7	x_4y_7	x_5y_7	x_6y_7	x_7y_7	x_8y_7	
							x_1y_6	x_2y_6	x_3y_6	x_4y_6	x_5y_6	x_6y_6	x_7y_6	x_8y_6		
						x_1y_5	x_2y_5	x_3y_5	x_4y_5	x_5y_5	x_6y_5	x_7y_5	x_8y_5			
					x_1y_4	x_2y_4	x_3y_4	x_4y_4	x_5y_4	x_6y_4	x_7y_4	x_8y_4				
				x_1y_3	x_2y_3	x_3y_3	x_4y_3	x_5y_3	x_6y_3	x_7y_3	x_8y_3					
			x_1y_2	x_2y_2	x_3y_2	x_4y_2	x_4y_2	x_5y_2	x_6y_2	x_8y_2						
		x_1y_1	x_2y_1	x_3y_1	x_4y_1	x_4y_1	x_5y_1	x_6y_1	x_8y_1							
	1								1							
PRODUCT	p_1	p_2	p_3	p_4	p_5	p_6	p_7	p_8	p_9	p_{10}	p_{11}	p_{12}	p_{13}	p_{14}	p_{15}	p_{16}

Figure 1.3: Mixed-operand Partial Products matrix, $n = 8$. Y is n bit two's complement fractional number, hence y_1 has weight equal to -2^{-1}.

1.2 Full-width Multiplier

A Full-width digital $n \times n$ multiplier computes the $2n$ bits output as a weighted sum of partial products. A parallel-tree multipliers reduces the matrix of partial products to two rows using a combination of full-adders and half-adders. The remaining two rows are then added by a carry propagate adder to produce the final result. In this Section at first the most common used techniques, Wallace and Dadda trees, are described; then details on an efficient VLSI implementation, the array multipliers, are given. Finally Sec. 1.2.4 describes Three Reduction Methods (TDM), a method for the generation of a parallel multiplier, optimized for speed, which will be used in the implementation of the proposed truncated multipliers.

1.2.1 Wallace-Tree Multiplier

The reduction scheme published by Wallace [2] begins by grouping the partial-product matrix into sets of three rows. Each set is reduced to two rows using half-adders on sets of two bits and full-adders on sets of three bits. Excess rows that do not belong to a set of three are passed to the next reduction stage unmodified. Each reduction stage is processed in a similar way until only two rows remain. Then a final CPA (Carry Propagate Adder) is used. Fig. 1.4 shows the dot diagram illustrating Wallace reduction for an 8×8 multiplier. Dot diagrams, developed by Dadda, are a convenient means for visualizing

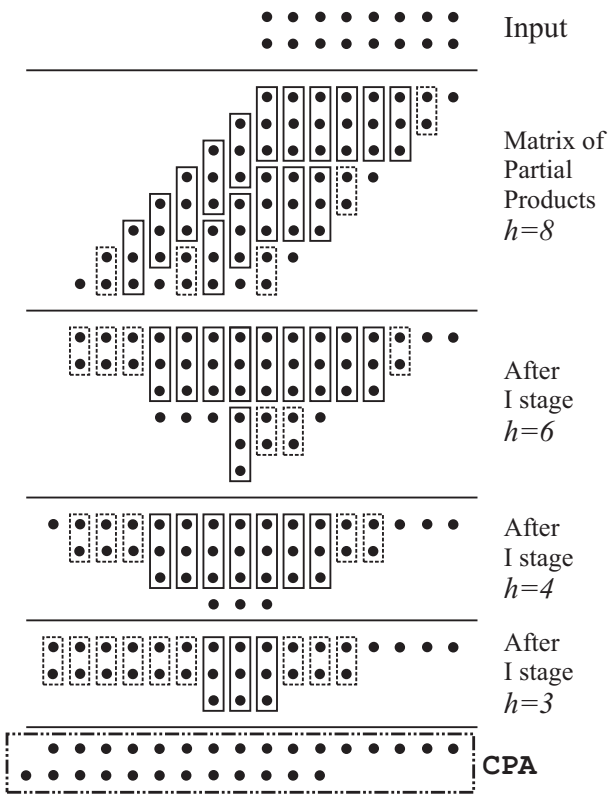

Figure 1.4: Wallace reduction for an 8×8 multiplier. Full circle: Full Adder. Dashed circle: Half Adder.

the placement of full-adders and half-adders. In such diagrams, dots represent bits, given by partial products. For example, the upper-right dot in the multiplication matrix is $x_8 y_8$. The circle with three dots represents a Full-Adder (FA). In the next stage the circle is substituted by its outputs: a bit sum, a dot in the same column, and a bit carry, a dot in the column on the left. Same reasoning is valid for the half-adder, represented by a circle with dashed line. Summarizing in the Wallace tree the number of the operands is reduced at the earliest opportunity, so, if there are m dots in the column we immediately apply $\lfloor m/3 \rfloor$ full adder to that column. This tends to minimize the overall delay by making the final CPA as short as possible.

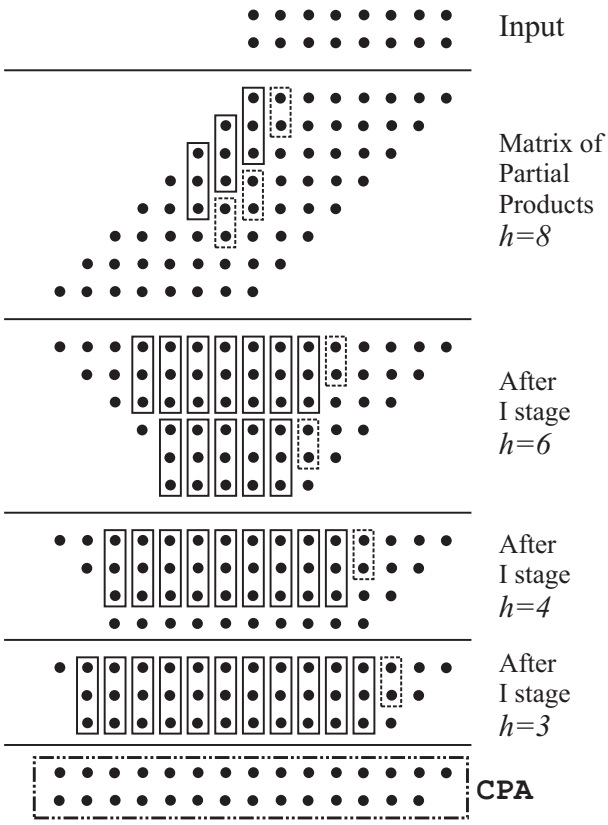

Figure 1.5: Dadda reduction for an 8×8 multiplier. Full circle: Full Adder. Dashed circle: Half Adder.

1.2.2 Dadda-Tree Multiplier

Wallace's method is refined by Dadda [3] into a technique that is optimum in the sense that it uses a minimum number of full-adders. At each stage of reduction, the height h of the matrix of Partial Products can be reduced following the Dadda's series $\{d_1, d_2, \ldots, d_j, \ldots\} = \{2, 3, 4, 6, 9, \ldots\}$, where:

$$\begin{aligned} d_1 &= 2 \\ d_{j+1} &= \left\lfloor \frac{3}{2} d_j \right\rfloor \end{aligned} \tag{1.13}$$

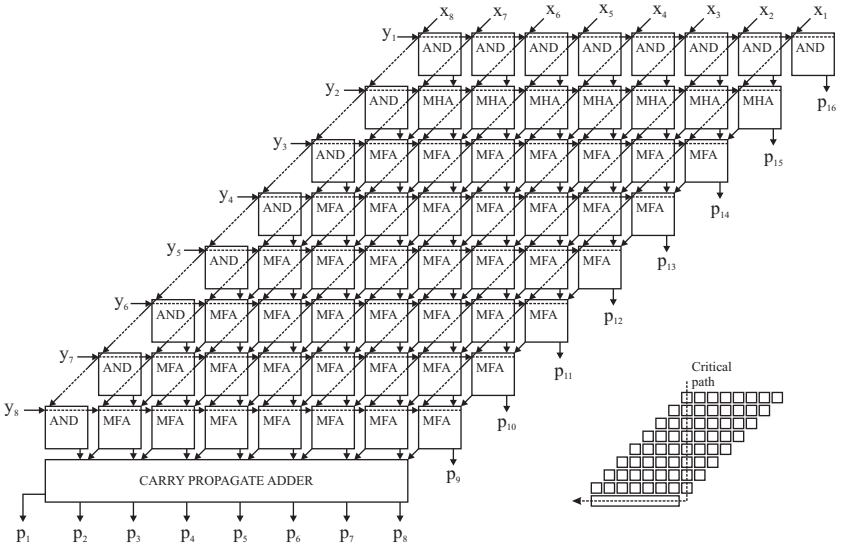

Figure 1.6: Unsigned array multiplier, $n = 8$.

Dadda's reduction uses the full-adder and the half-adder only if the height of the matrix can be reduced following Dadda series, until 2 rows are reached. Fig. 1.5 shows Dadda reduction for an 8×8 multiplier. The partial-product matrix has a maximum height of 8, so the first step of reduction leads to $h = 6$, then $4, 3, 2$; four reduction steps are necessary.

Dadda reduction of an $n \times n$ multiplier requires $\left(n^2 - 2n + 3\right)$ full-adders, $(n-1)$ half-adders, and a carry-propagate adder with a length of $(2n-2)$ [4]. Dadda reduction uses fewer full-adders and half-adders than Wallace reduction, but requires a longer carry-propagate adder.

1.2.3 Array Multiplier

Array multiplier is a very common type of parallel multiplier. Array multipliers are not as fast as tree multipliers, but their regular structure and local interconnect are advantages for very-large-scale-integration (VLSI) implementation.

Unsigned Array Multipliers Fig. 1.6 shows an 8×8 unsigned array multiplier. Each cell in the array receives a bit from the multiplicand, x_i, and a

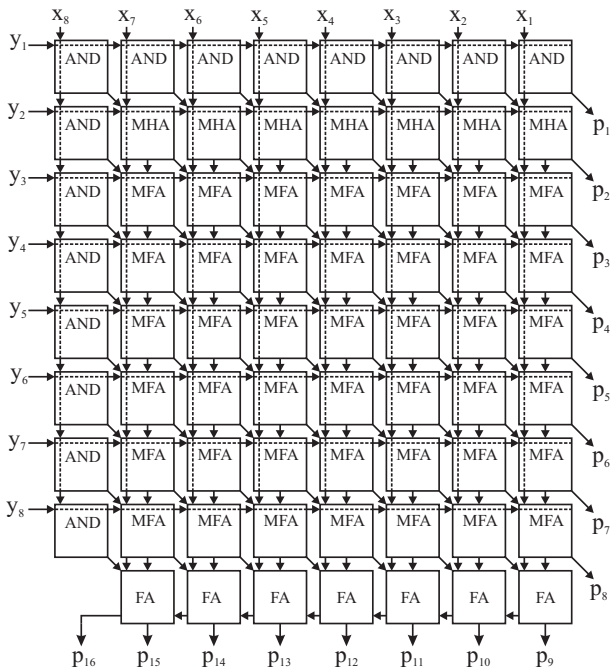

Figure 1.7: Unsigned array multiplier, $n = 8$, rectangular shape.

bit from the multiplier, y_j. Cells comprising the top row and left diagonal are *AND* gates, which generate the unsigned partial-product bits as described in Sec. 1.1.1 and pass them to modified half-adders (*MHA*) or modified full-adders (*MFA*). A *MHA* is a half-adder with an additional *AND* gate that generates $x_i y_j$, and adds it to a single bit generated by another block. A *MFA* is a full-adder with an additional *AND* gate that generates $x_i y_j$, and adds it to the sum bit and the carry bit generated by two *MHAs* or *MFAs* above it. The least-significant bits of the product are produced directly by cells in the array. The most-significant bits in the product are produced by a carry-propagate adder that adds the sum bits and carry bits from the bottom row of cells in the array. The critical path of an array multiplier, shown in the lower right of Fig. 1.6, is through the highest column in the array plus the carry-propagate adder. Fig. 1.7 shows the array multiplier redrawn as it typically appears in the literature. In this figure, a ripple-carry adder is used as the final carry-propagate adder (CPA), although any CPA could be used.

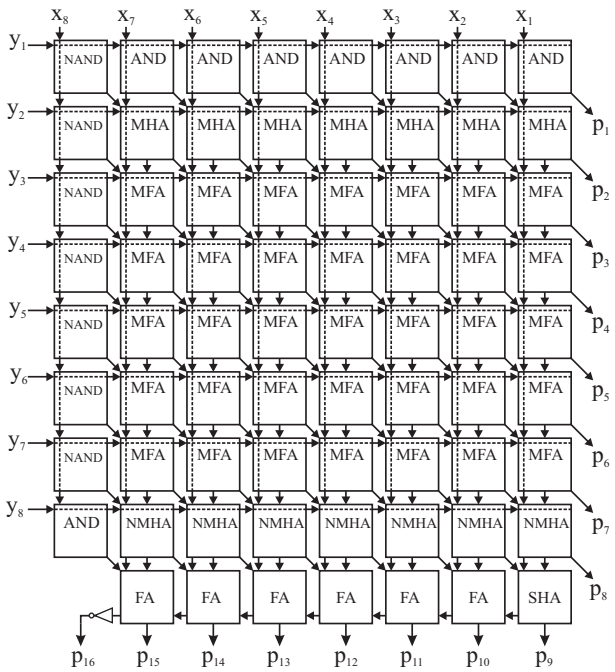

Figure 1.8: Signed array multiplier, $n = 8$.

Two's-Complement Array Multipliers Fig. 1.8 shows an 8×8 array multiplier modified for two's-complement operation. As described in Sec. 1.1.2, some of the bits in the partial-product matrix must be inverted, and constant 1 must be added. Half of the bit inversions are performed by replacing *AND* gates with *NAND* gates. The remaining bit inversions are accomplished by replacing *MFAs* with *NMFAs*. A *NMFA* is a negating-modified full-adder, which is simply a *MFA* with a *NAND* gate to generate $\overline{x_i y_j}$ instead of an and gate generating $x_i y_j$. As the both operands are *n* bits long, 1 must be added to the 2^{2n-1} and 2^n columns. Adding 1 to the 2^{2n-1} column is accomplished by inverting the p^{2n-1} bit. Adding 1 to the 2^n column is accomplished by setting the carry into the carry-propagate adder to 1. If a ripple-carry adder is used, as in Fig. 1.8, then the half-adder can be replaced by a specialized half-adder (*SHA*). Specialized half-adders add two bits plus a constant 1.

1.2.4 Three Dimensional Minimization (TDM) method

In [5] Oklobdzija *et al.* suggested a new approach, the Three Dimensional Method (TDM) for Partial Product Reduction Tree (PPRT). The authors took the approach of generation the compressors of maximum possible size (i.e., the size of the multiplier): they abandoned the notion of levels and undertook a design of an optimized one-level compressor which evolved into an optimization process involving the entire array. The method is based on the fact that not all the inputs and the outputs of a compressor contribute equally to the delay. Therefore, Oklobdzija *et al.* sort them in a way which favors the use of fast inputs and outputs in the path that are critical to the speed, while they assign slow inputs to the signal paths which belong to the domain where an increase in the delay is tolerable. The algorithm for Automatic Generation of Partial Product Array can be read in [5]. Even if the number of cells used by TDM approach and Dadda' one is the same, the two approaches are very different since in TDM algorithm all the partial products are compressed into a single step, hence no intermediate partial products are considered in TDM approach. However, TDM still produces a carry save number to be translated to a conventional form with a fast carry propagate adder (CPA).

Oklobdzija *et al.* suggested a good heuristic for finding the optimal PPRT (Partial Product Reduction Tree), but no proof about the performance of this heuristic are given. Stelling *et al.* [6] provides a formal characterization of optimal PPRT circuits and prove a number of properties about them. Furthermore the authors present an algorithm that produces a minimum delay circuit in time linear with the size of the inputs, providing tight lower bounds on multiplier circuit delays. These results are combined to create a program that finds optimal TDM multiplier designs. Using the program, Stelling *et al.* show that, while the heuristic used in [5] does not always find the optimal TDM circuit, it performs very well in terms of overall PPRT circuit delay and they found, with their new search algorithms, a better PPRT circuits for reducing the delay of the entire multiplier.

In [7] the authors propose a new algorithm for synthesizing fast arithmetic circuits. Analyzing the Wallace tree compressor (i.e. bit-level carrysave addition array) the authors note that the scheme has been applied only in a rather restrictive way, i.e. for implementing fast multipliers and for generating fixed structures without considering the characteristic of the input signals. Since the Wallace algorithm is implemented by two stage, the carry-save additions of the partial products (implemented with full-adders (FAs)) and a carry-propagate addition (CPA) to sum the two addends produced by the first stage, in [7] the

authors focused on the optimization of the first stage. They looked for a solution to the problem of reducing the addend matrix to a matrix with at most two addends at each column, by allocating FAs. They proposed a simple constructive procedure for allocating FAs to reduce the single column with m addends to one with at most two addends. The algorithm can be found in [7].

1.3 Truncated Multiplier

A truncated multiplier is an $n \times n$ multiplier with n bits output. As it is shown if Fig. 1.9 the partial products can be divided into two subsets. The least significant part (LSP) includes the n less significant columns of the partial product matrix, while the most significant part (MSP) includes the remaining columns. The full-width multiplier output, P is given by

$$P = S_{\text{MSP}} + S_{\text{LSP}} \tag{1.14}$$

where S_{MSP} and S_{LSP} represent the weighted sum of the elements of MSP and LSP respectively.

When a n bits output is needed, the most accurate choice is using the full rounded multiplier: it computes all the matrix of partial products, add a constant to the result on $2n$ bits and takes only the first n bits of the sum. The error introduced by the full rounded multiplier is calculated in Sec. 1.3.1. Unfortunately the full rounded multiplier is the solution with the highest area occupation and power dissipation.

A second possibility is using a truncated multiplier in which the partial products of the LSP are discarded assuming that their contribution to the n most significant bits of the output is negligible. This solution is very advantageous in terms of hardware performances. For example in Fig. 1.10 there is the implementation of the array truncated multiplier with $n = 8$. The cells for the LSP matrix are not present and the final circuit halves the number of cells compared to the full-width one (Fig. 1.7). However, a straightforward analysis shows that the direct elimination of the partial products of the LSP causes a very big error bounded by $(n/2 - 1)\mathbf{lsb}$, where $\mathbf{lsb}$ is the weight of the least significant bit of the result.

Ranging between these two extreme cases, it is possible to devise numerous circuit alternatives: only some of some of the partial products in the LSP are removed to trade-off accuracy with hardware cost. Several techniques that follow this idea have been proposed in the Literature [8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19]. These techniques introduce suitable compensation circuits,

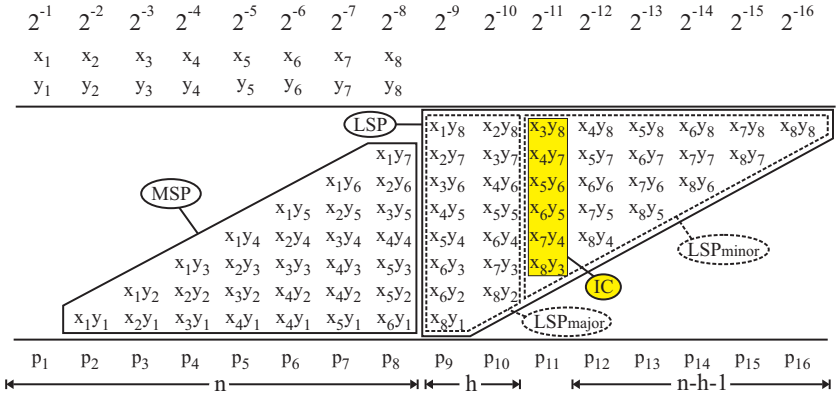

Figure 1.9: Subdivision of the matrix of Partial Products, for unsigned multiplier, $n = 8$. MSP (Most Significant Part) of the PPs Matrix is constituted by the first n columns. LSP (Less Significant Part) of the PPs Matrix is constituted by the last n columns. IC (Input Correction) vector is the $(n + h + 1)^{th}$ column.

that partly compensates for the dropped terms, thereby reducing the approximation error. The proposed approaches differ one another in the choice and the implementation of the compensation circuit, but they can be classified into two groups, constant correction methods and variable correction methods. To discuss these techniques, let's indicate the *h* most significant columns of LSP as $\text{LSP}_{\text{major}}$, while the remaining $n_{eq} = n-h$ columns will be named $\text{LSP}_{\text{minor}}$ (see Fig. 1.9). The value of *h* is a design parameter, that can range from $h = 0$ to $h = n$. In addition, the leftmost column of $\text{LSP}_{\text{minor}}$ (highlighted in yellow in Fig. 1.9) is named Input Correction (IC). The IC, composed by n_{eq} partial products, is used to estimate the weighted sum of the elements of the $\text{LSP}_{\text{minor}}$. For unsigned multiplier and signed multiplier with $h \neq 0$, IC is given by:

$$IC = \{x_{h+1}y_{n+1-i}\}, i = 1 \dots n_{eq} \tag{1.15}$$

while the first and the last elements of IC are complemented if a signed multiplier with $h = 0$ is considered ($\overline{x_1 y_n}, \overline{x_n y_1}$ instead of $x_1 y_n, x_n y_1$).

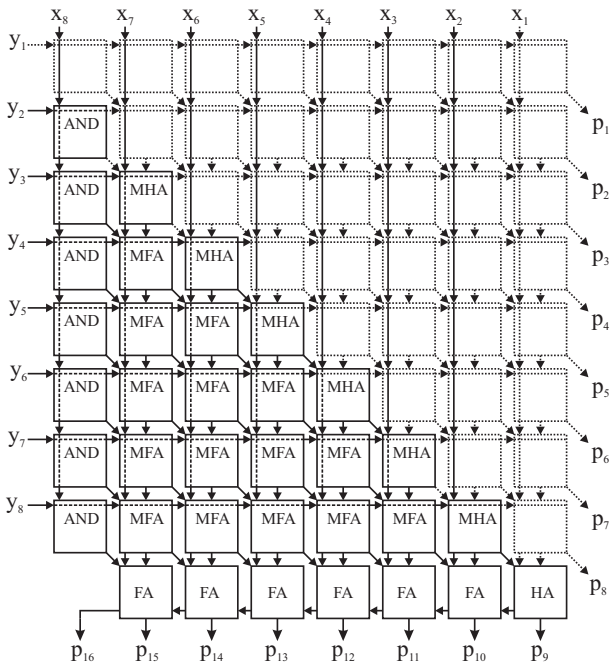

Figure 1.10: Unsigned array truncated multiplier, $n = 8$. Dashed lines are for the not computed cells.

1.3.1 Full Rounded (Round-to-Nearest) Multiplier

The output of the full-rounded multiplier is:

$$P_{round} = trunc_n(S_{\text{MSP}} + S_{\text{LSP}} + K_{\text{round}}) \tag{1.16}$$

where $trunc_j$ indicates the truncation to j bits of the result, and K_{round} is the rounding constant, whose value is $\mathbf{lsb}/2$. Fig. 1.11 shows the partial product matrix of 8×8 unsigned rounded multiplier. The full-rounded multiplier gives the smallest possible error $e = P - P_t$, bounded by $\mathbf{lsb}/2$. In the hypothesis of input bits independent and identically distributed the mean and mean square value of the error are:

$$\mu_{\text{round}} = E[e_{round}] = 0 \tag{1.17}$$

$$\varepsilon^2_{\text{round}} = E[e^2_{round}] = \frac{1}{12}\mathbf{lsb}^2 \tag{1.18}$$

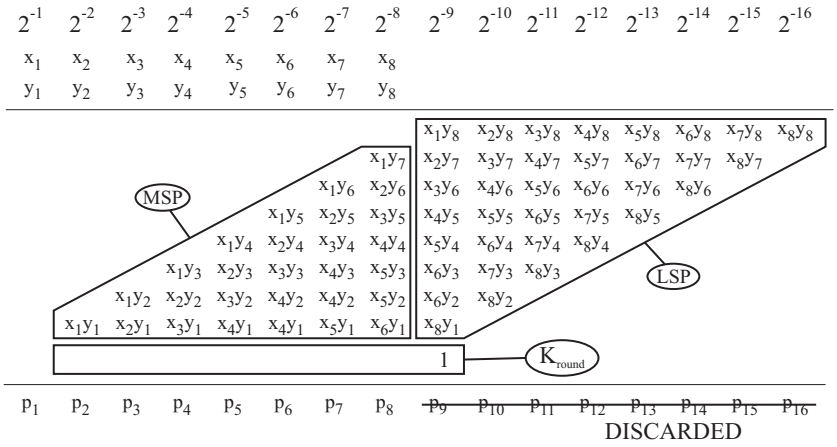

Figure 1.11: Full-rounded multiplier, $n = 8$. K_{round} is the rounding constant. The *n* less significant bits of the results are discarded.

In Literature it is common to refer to this multiplier in order to make a comparison in terms of area, delay, power and error. The comparison can be interesting when we speak about hardware performances, but it's worth noticing that $\varepsilon^2_{\text{round}}$ is a value which can't be reached by any truncated multiplier. As it will show in Sec. 2.3.1 the real lower bound, called intrinsic error, is higher than $\varepsilon^2_{\text{round}}$.

1.3.2 Constant Correction Methods

The Constant Correction Methods (CCM) use a constant value, independent on the actual values of the inputs, in order to estimate the $\text{LSP}_{\text{minor}}$. The multiplier output can be written as:

$$P_{CCM} = trunc_n(S_{\text{MSP}} + S_{\text{LSP}_{\text{major}}} + constant) \tag{1.19}$$

where $S_{\text{LSP}_{\text{major}}}$ is the weighted sum of the elements of the $\text{LSP}_{\text{major}}$.

The simplest approach has been proposed by Kidambi *et. al.* in [8]. In this technique the LSP is eliminated and is substituted by a constant term, calculated considering only the lose carries. This approach reduces up to 50% the area of the full-width multiplier, but introduces a rather large error, which rapidly increases with *n*, resulting impractical in most applications.

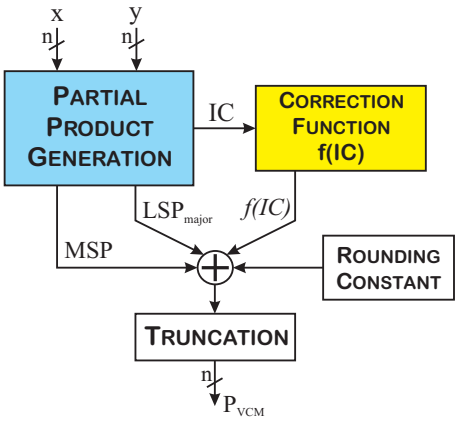

Figure 1.12: General scheme of a truncated multiplier. The compensation function $f(\mathrm{IC})$ is estimated from the value of IC. The $\mathrm{LSP}_{\mathrm{minor}}$ is not computed. The result is reported on n bits with a truncation operation.

In [9] Lim proposes a technique to improve the accuracy by discarding only the $\mathrm{LSP}_{\mathrm{minor}}$, shown in Fig. 1.9. The authors noticed that the major contribution to the error was given by the discarded carries of $\mathrm{LSP}_{\mathrm{minor}}$ which ripples into $\mathrm{LSP}_{\mathrm{major}}$. Hence in order to calculate the constant value, Lim only estimates these carries. The value of h is used as a parameter to trade-off accuracy with complexity. The approach of [9] is refined in [11] where the rounding error is also taken into account in the computation of the optimal value for the correction constant.

1.3.3 Variable Correction Methods

The accuracy of truncated multipliers can be significantly improved using variable-correction truncated multipliers [18, 10, 11, 12, 13, 14, 15, 16, 17, 19], that compensate the effect of the dropped terms with a non-constant compensation function. The terms in the IC are used to estimate the weighted sum of the elements of the $\mathrm{LSP}_{\mathrm{minor}}$ and the multiplier output is computed as:

$$P_{VCM} = trunc_n(S_{\mathrm{MSP}} + S_{\mathrm{LSP}_{\mathrm{major}}} + f(\mathrm{IC}) + K_{\mathrm{round}}) \tag{1.20}$$

where $f(\mathrm{IC})$ is a suitable compensation function. The Fig. 1.12 shows the general scheme of the truncated multiplier designed using a variable-correction

scheme. $f(\mathrm{IC})$ and the rounding constant are added to MSP and $\mathrm{LSP}_{\mathrm{major}}$ to obtain the result, which is finally reported on *n* bits with a truncation operation. For a given h value, the hardware complexity and approximation error introduced by the scheme in Fig. 1.12 depends on the choice of the compensation function $f(\mathrm{IC})$.

King, Stine and Park Methods

In [10] King *et al.* propose to add the partial products of the IC in the rightmost column of the $\mathrm{LSP}_{\mathrm{major}}$. This is a simple but very effective way to correlate the correction value to the $\mathrm{LSP}_{\mathrm{minor}}$. Combining [11], constant correction methods, and [10], variable correction methods, in [12] Stine *et al.* propose a so-called hybrid correction method, where only a subset of the IC elements are summed in the rightmost column of the $\mathrm{LSP}_{\mathrm{major}}$. In [20] Park *et al.* improve the approach of [10] by summing in the rightmost column of the $\mathrm{LSP}_{\mathrm{major}}$ all the IC terms and a further correcting bit.

Jou et al. Method

In [13] a correction function for truncated multipliers is proposed by Jou *et al.*, by manipulating the partial products in IC with a circuit composed by *AND-OR* gates. The authors propose two correction functions, depending on the multiplier.

For *signed multiplier* with $h = 0$,

$$\alpha_{\mathrm{JOU}} = \overline{\bigvee_{i=1}^{n} x_i y_{n+1-i}} = \bigwedge_{i=1}^{n} \overline{x_i y_{n+1-i}} \tag{1.21}$$

$$f_{\mathrm{JOU}}(\mathrm{IC}) = \left\lceil \sum_{i=2}^{n-1} x_i y_{n+1-i} + \alpha_{\mathrm{JOU}} \right\rceil 2^{-n} \tag{1.22}$$

where $\bigvee$ and $\bigwedge$ represent the Boolean operator *OR* and *AND* operator respectively. Note that α_{JOU} is a constant added only when all the elements of the IC are equal to zero, with the exception of the first and the last elements, which should be equal to 1.

For *signed ($h \neq 0$) and unsigned multiplier*

$$\alpha_{\text{JOU}i} = x_i y_{n+h+1-i} \wedge \left(\bigvee_{j=h+1}^{i-1} x_j y_{n+h+1-j} \right) \quad i = h+2, \ldots, n \tag{1.23}$$

$$f_{\text{JOU}}(IC) = \left[\sum_{i=h+2}^{n} \alpha_{\text{JOU}i} \right] 2^{-n-h} \tag{1.24}$$

The technique in [13] is affected by a significant mean and mean square error. The circuit that calculates $f(\text{IC})$ is characterized by a slow ripple architecture and the number of output bits is equal to $n + h$ instead of n.

Curticapean et al. Method

Curticapean *et al.* [14], use a modified version of the correction circuit of [13] suitable for unsigned multipliers. The proposed correction function is:

$$\alpha_{\text{CCP}i} = x_i y_{n+h+1-i} \wedge \left(\bigvee_{j=h+1}^{i-1} x_j y_{n+h+1-j} \right) \quad i = h+2, \ldots, n-1 \tag{1.25}$$

$$\alpha_{\text{CCP}n} = \bigvee_{i=h+1}^{n} x_i y_{n+h+1-i} \tag{1.26}$$

$$f_{\text{CCP}}(IC) = \left[\sum_{i=h+2}^{n} \alpha_{\text{CCP}i} \right] 2^{-n-h} \tag{1.27}$$

Note that the function is proposed only for unsigned multiplier. This technique provides good error performances but the compensation function is still based on a slow ripple architecture. Furthermore in [14] no explanation is given to justify the improved error performances.

Van et al. Method

In [15] Van *et al.* propose a truncated signed multiplier architecture in which the correction function proposed in [13] is generalized by considering either the IC terms or their complements. The optimal pattern of inversions applied

to the partial products in IC is obtained through exhaustive simulation. The proposed correction function, for signed multiplier with $h = 0$ is:

$$\alpha_{\text{VANO}} = \overline{\overline{x_1 y_n} \wedge \left(\bigwedge_{i=2}^{n-1} x_i y_{n+1-i} \right) \wedge \overline{x_n y_1}} \tag{1.28}$$

$$= x_1 y_n \vee \left(\bigvee_{i=2}^{n-1} \overline{x_i y_{n+1-i}} \right) \vee x_n y_1 \tag{1.29}$$

$$f_{\text{VANO}}(IC) = \left[\sum_{i=2}^{n-1} x_i y_{n+1-i} + \alpha_{\text{VANO}} \right] 2^{-n} \tag{1.30}$$

The technique has been extended to the case $h > 0$ in [16]:

$$\alpha_{\text{VAN}} = \left(\bigvee_{i=h+1}^{n-1} \overline{x_i y_{n+h+1-i}} \right) \wedge x_n y_{h+1} \tag{1.31}$$

$$f_{\text{VAN}}(IC) = \left[\sum_{i=h+1}^{n-1} x_i y_{n+h+1-i} + \alpha_{\text{VAN}} \right] 2^{-n-h} \tag{1.32}$$

In [15] and [16] $f(\text{IC})$ is still implemented with a slow ripple architecture. Furthermore the optimal value of the function $f(\text{IC})$ is computed through an exhaustive search. Hence the effectiveness of these techniques is verified only for $n \leq 16$ and $h = 0, 1, 2$. The result obtained by Van is explained in [17]. The authors investigate the dependency of the carry bits on the partial products and the inputs, and propose three carry estimation schemes. Since the error performance of [17] does not improve [15], [16] the multipliers proposed in [17] will not be considered in the following.

Kuang et al. Method

In [18] the authors propose a minimal modification of the technique presented by Strollo *et al.* [21]. This correction, done in order to obtain a lower mean square error or mean error, has been obtained thought a brute force computation. Hence it is valid only for the analyzed case, unsigned multiplier with $h = 0$. The authors don't explain how the technique can be extended and what happen if we consider a signed multiplier.

The correction proposed by Strollo *et al.* in [21] is

$$
\begin{aligned}
c_1 &= x_n y_1 \oplus x_{n-1} y_2 \oplus x_2 y_{n-1} \\
c_2 &= (x_n y_1 \wedge x_{n-1} y_2) \vee x_2 y_{n-2} \wedge (x_n y_1 \oplus x_{n-1} y_2) \oplus x_1 y_n \\
c_3 &= (x_n y_1 \wedge x_{n-1} y_2) \vee x_2 y_{n-2} \wedge (x_n y_1 \oplus x_{n-1} y_2) \wedge x_1 y_n
\end{aligned}
$$

$$
\begin{aligned}
\alpha_{\text{STROLLO}} &= c_1 + c_2 + c_3; \\
f_{\text{STROLLO}}(IC) &= \left\lceil \sum_{i=3}^{n-2} x_i y_{n+1-i} + \alpha_{\text{STROLLO}} \right\rceil 2^{-n}
\end{aligned}
\tag{1.33}
$$

Kuang *et al.* notice that c_1, c_2, c_3 can't be 1 simultaneously. Therefore in [18] the standard addition is modified in order to generate only two outputs such that the retained adder cells can also be slightly simplified. The authors propose two different configurable error-compensation circuit.

New I, which provides a lower variance but an higher mean error:

$$
\begin{aligned}
c_1 &= \overline{\overline{x_1 y_n} \vee \overline{x_2 y_{n-1}}} \vee \overline{\overline{x_{n-1} y_2} \vee \overline{x_n y_1}} \\
c_2 &= \overline{\overline{x_1 y_n} \wedge \overline{x_2 y_{n-1}}} \vee \overline{\overline{x_{n-1} y_2} \wedge \overline{x_n y_1}} \\
\alpha_{\text{NEWI}} &= c_1 + c_2; \\
f_{\text{NEWI}}(IC) &= \left\lceil \sum_{i=3}^{n-2} x_i y_{n+1-i} + \alpha_{\text{NEWI}} \right\rceil 2^{-n}
\end{aligned}
\tag{1.34}
$$

New II, which provides a lower mean error with an higher variance:

$$
\begin{aligned}
c_1 &= \overline{\overline{x_1 y_n} \vee \overline{x_2 y_{n-1}}} \vee \overline{\overline{x_{n-1} y_2} \vee \overline{x_n y_1}} \\
c_2 &= \overline{\overline{x_1 y_n} \wedge \overline{x_2 y_{n-1}}} \vee \overline{\overline{x_{n-1} y_2} \wedge \overline{x_n y_1}} \\
c_3 &= \overline{x_3 y_{n-3}} \wedge \overline{x_{n-3} y_3} \\
\alpha_{\text{NEWII}} &= c_1 + c_2 + c_3; \\
f_{\text{NEWII}}(IC) &= \left\lceil \sum_{i=3}^{n-2} x_i y_{n+1-i} + \alpha_{\text{NEWII}} \right\rceil 2^{-n}
\end{aligned}
\tag{1.35}
$$

It is important to underline that the authors can obtain a lower but an higher mean and viceversa. In the following it will be demonstrated that both parameters are important and should be kept as lower as possible together.

Michard et al. Method

The approach proposed in [19] is a noteworthy exception with respect to the scheme of Fig. 1.12. The prediction-selection method of [19] is based on a logical computation of the values to be added to S_{MSP} and to $S_{\text{LSP}_{\text{major}}}$ (prediction), followed by a simplification process (selection). Results are presented for truncated multipliers with $n = 8, 12,$ and 16. The approach of [19] is not applicable to higher n values, due to the fast growing computational cost of the prediction process.

1.4 Conclusion

From the Literature survey it comes out that the constant correction techniques [8, 9, 11] are the simplest to be implemented. Moreover, their error properties can be described analytically rather easily, even if it will not be done in this dissertation. Unfortunately, the constant correction techniques are also the less effective in terms of approximation error.

On the other hand, the variable correction techniques proposed to date, while significantly improving accuracy, show substantial limits. In particular, they are not derived from an analytical theory but rather heuristically or with the help of exhaustive searches. Thus, in many cases the proposed techniques can not be applied to multipliers with large bit widths (say, 24 or 32 bits) and/or can not be considered for a possible implementation in automatic synthesis tools. In addition, the error performances are customarily computed numerically through exhaustive simulations. This approach can be pursued only for small n values since the simulation time increases as $O(2^{2n})$, requiring an unreasonable amount of CPU time when n increases. As an example, the technique proposed in [16] is only verified for $n \leq 16$ and $h = 0, 1, 2$.

Chapter 2

LMS Truncated Multiplier

The optimal compensation function $f_{opt}(\mathrm{IC})$, that minimizes the mean square error of the truncated multiplier, is analytically calculated in this chapter. In Sec. 2.3 it is shown that $f_{opt}(\mathrm{IC})$ is a quadratic form of the elements in IC and that the corresponding minimum mean square error is the lower error bound for any truncated multiplier.

With the help of the developed theory, the derivation of a sub-optimal compensation function, $f_{lin}(\mathrm{IC})$, expressed as a linear combination of the elements in IC, is described in Sec. 2.4. The linear compensation function yields error performances very similar to those achievable using $f_{opt}(\mathrm{IC})$ and has the additional advantage of easy hardware implementation. The aspects related to the quantization of the coefficients of $f_{lin}(\mathrm{IC})$ are investigated in Sec. 2.5. Hence it is given the expression, in closed form, of the compensation function, Linear Minimum mean Square error (LMS compensation function). Finally in Sec. 2.8 it is shown how the results, obtained for an unsigned multiplier can be extended to signed and mixed multipliers. The hardware implementation and the performances of the sub-optimal linear compensation function are described in the next chapter.

2.1 Definitions and Assumptions

In the following we will consider unsigned multipliers. Without loss of generality, we will moreover assume that the inputs of the multiplier represent

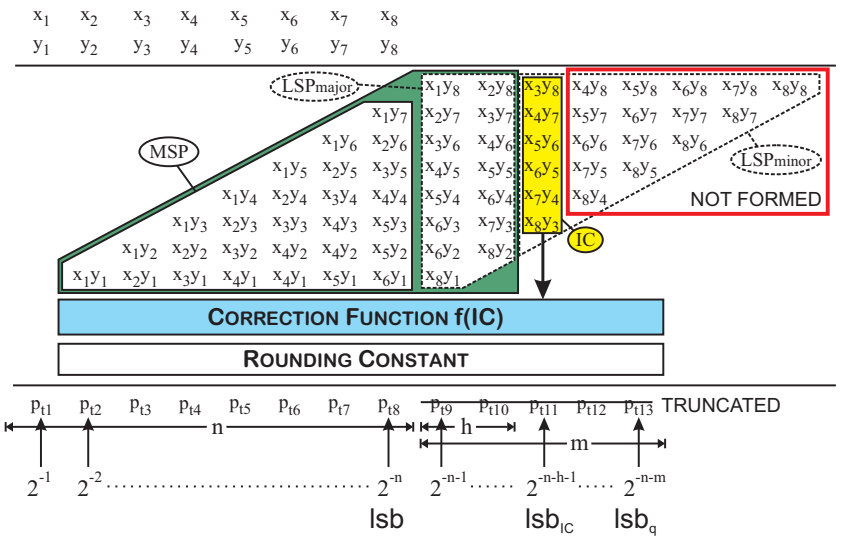

Figure 2.1: Partial Products matrix for unsigned truncated multiplier, $n = 8$ and $h = 2$, with variable-correction scheme.

fractional values in $[0, 1)$:

$$x = \sum_{i=1}^{n} x_i 2^{-i}; \quad y = \sum_{i=1}^{n} y_i 2^{-i} \tag{2.1}$$

Each bit x_i and y_i is assumed to be independent and uniformly distributed, with a probability $1/2$ of being one.

We focus on variable correction methods whose configuration, for $n = 8$ and $h = 2$, is represented in Fig. 2.1. The $\text{LSP}_{\text{minor}}$ is not used and is estimated with the elements of IC. The result is given by the sum of the compensation function $f(\text{IC})$, the rounding constant K_{round}, MSP and $\text{LSP}_{\text{major}}$, truncated on n bits. Note that:

$$\mathbf{lsb} = 2^{-n} \text{ is the less significant bit of the result;} \tag{2.2}$$

$$\mathbf{lsb}_{IC} = 2^{-n-h-1} \text{ is the weight of the IC column;} \tag{2.3}$$

$$\mathbf{lsb}_{q} = 2^{-n-m} \text{ is the less significant bit of } f(\text{IC}) \text{ and } K_{\text{round}}. \tag{2.4}$$

The elements of the IC (see Fig. 2.1) will be indicated as follows:

$$IC = \left[\gamma_1, \gamma_2, \ldots, \gamma_{n_{eq}}\right] \tag{2.5}$$

where:

$$n_{eq} = n - h \tag{2.6}$$

and:

$$\gamma_i = x_{h+i} y_{n+1-i} \tag{2.7}$$

Let's define Θ the set of all the possible values of the vector IC ($2^{n_{eq}}$ elements) and $\Omega(A)$ the subset of the set of all couples (x, y) which gives $IC = A$. For instance, in the multiplier of Fig. 2.1, the subset $\Omega([1,1,1,1,1,1])$ is composed by 16 possible couples (x, y):

$$(x_{8..1}\ ;\ y_{8..1}) = (1,1,1,1,1,1,-,-\ ;\ 1,1,1,1,1,1,-,-) \tag{2.8}$$

while the subset $\Omega([1,1,0,1,1,1])$ is composed by 48 (x, y) values:

$$\begin{aligned}
(x_{8..1}\ ;\ y_{8..1}) &= (1,1,1,0,1,1,-,-\ ;\ 1,1,0,1,1,1,-,-) \\
(x_{8..1}\ ;\ y_{8..1}) &= (1,1,1,1,1,1,-,-\ ;\ 1,1,0,1,1,1,-,-) \\
(x_{8..1}\ ;\ y_{8..1}) &= (1,1,1,0,1,1,-,-\ ;\ 1,1,1,1,1,1,-,-)
\end{aligned} \tag{2.9}$$

where the symbol "$-$" indicates a don't care value (the variable can be either 1 or 0).

As the elements of IC are formed by partial products obtained by an *AND* operator, in general the number $n_{xy}(A)$ of (x, y) values in a set $\Omega(A)$ depends by nz, the number of zeros in the IC: $n_{xy}(A) = 3^{nz} \cdot 2^{2h}$. For the same reason the probability of IC to be equal to A depends on the number of zeros in A: $P(IC = A) = n_{xy}(A) \cdot 2^{-2n} = 3^{nz} \cdot 2^{-2n_{eq}}$. For instance, in the multiplier of Fig. 2.1: $P([1,1,0,1,1,1]) = 48/2^{16} = 3 \times 2^{-12}$.

The weighted sum of the elements in $\text{LSP}_{\text{minor}}$ is given by:

$$S_{\text{LSP}_{\text{minor}}} = \left(2^{-n-h-1} \sum_{i=1}^{n_{eq}} \gamma_i \right) + \left(\sum_{i=h+2}^{n} \sum_{j=n+h+2-i}^{n} x_i y_j 2^{-i-j} \right) \tag{2.10}$$

The mean value of $\text{LSP}_{\text{minor}}$ in $\Omega(A)$ is:

$$\mu_{\text{LSP}}(A) \triangleq \underset{(x,y)\in\Omega(A)}{E} \left\{ S_{\text{LSP}_{\text{minor}}}(x, y) \right\} \tag{2.11}$$

while the variance of $\text{LSP}_{\text{minor}}$ in $\Omega(A)$ is:

$$\sigma^2_{\text{LSP}}(A) \triangleq \underset{(x,y)\in\Omega(A)}{E} \left\{ \left(S_{\text{LSP}_{\text{minor}}}(x, y) - \mu_{\text{LSP}}(A) \right)^2 \right\} \tag{2.12}$$

where $\underset{(x,y)\in\Omega(A)}{E}\{\alpha\}$ is the mean conditioned to all couples $(x, y) \in \Omega(A)$. In order to simplify the notation in the following this mean will be indicated with:

$$\underset{IC=A}{E}\{\ldots\} \triangleq \underset{(x,y)\in\Omega(A)}{E}\{\ldots\} \tag{2.13}$$

2.2 Error Analysis for Truncated Multiplier

The error e_{total} introduced by the truncated multiplier of Fig. 2.1 is computed with respect to the full-width multiplier output. As P is the output of the full width multiplier and P_{t} the output of the truncated multiplier, looking at Fig. 2.1, it is evident that:

$$P = S_{\text{MSP}} + S_{\text{LSP}_{\text{major}}} + S_{\text{LSP}_{\text{minor}}} \tag{2.14}$$

$$P_{\text{t}} = S_{\text{MSP}} + f(\text{IC}) + K_{\text{round}} + e_{\text{trunc}} \tag{2.15}$$

where S_{MSP}, $S_{\text{LSP}_{\text{major}}}$ and $S_{\text{LSP}_{\text{minor}}}$ are the weighted sum of MSP, $\text{LSP}_{\text{major}}$ and $\text{LSP}_{\text{minor}}$ respectively, $f(\text{IC})$ is the compensation function, K_{round} the rounding constant and e_{trunc} represents the error due to the truncation operation (the output from $n + m$ bits is reported to n bits). Hence the error introduced by a truncated multiplier is given by:

$$e_{\text{total}} = P_{\text{t}} - P = f(\text{IC}) - S_{\text{LSP}_{\text{minor}}} + e_{\text{trunc}} + K_{\text{round}} \tag{2.16}$$

Aim of the work is to determine the optimal compensation function, $f_{opt}(\text{IC})$, that minimizes the mean square error:

$$\varepsilon^2_{\text{total}} = E\left\{e^2_{\text{total}}\right\} \tag{2.17}$$

The equation (2.16) can be rearranged as follows:

$$e_{\text{total}} = e_{\text{erasing}} + e_{\text{trunc}} + K_{\text{round}} \tag{2.18}$$

where:

$$e_{\text{erasing}} = f(\text{IC}) - S_{\text{LSP}_{\text{minor}}} \tag{2.19}$$

Since K_{round} is a constant chosen with the compensation function in order to minimize $\varepsilon^2_{\text{total}}$, two error sources affect the truncated multiplier. The first error source, e_{erasing}, is due to the use of the compensation function $f(\text{IC})$ in place of the sum of the elements in $\text{LSP}_{\text{minor}}$. The second error source, e_{trunc}, is due to the truncation of the result.

Indicating with σ^2_{total} and μ_{total} variance and mean value of e_{total}, the mean square error is given by:

$$\varepsilon^2_{\text{total}} = \sigma^2_{\text{total}} + (\mu_{\text{total}})^2 \tag{2.20}$$

where:

$$\mu_{\text{total}} = \mu_{\text{erasing}} + \mu_{\text{trunc}} + K_{\text{round}} \tag{2.21}$$

$$\sigma^2_{\text{total}} = \sigma^2_{\text{erasing}} + \sigma^2_{\text{trunc}} + 2\,\text{COV}(e_{\text{erasing}}, e_{\text{trunc}}) \tag{2.22}$$

and $\text{COV}(w,t)$ is the covariance between w and t. The equation (2.22) can be simplified if we assume that e_{trunc} is independent from e_{erasing} (hence the covariance is zero):

$$\sigma^2_{\text{total}} \simeq \sigma^2_{\text{erasing}} + \sigma^2_{\text{trunc}} \tag{2.23}$$

Note that (2.23) is not exact, since the bits discarded by the truncation operation depend on the compensation function $f(\text{IC})$ (see Fig. 2.1). Hence, the truncation error is not independent from the erasing error. Nevertheless we will shown in Sec. 2.5 that equation (2.23) represents a very good approximation of σ^2_{total}.

2.2.1 Statistical Properties of the Truncation Error (e_{trunc})

The truncation error is given by the weighted sum of truncated bits:

$$e_{\text{trunc}} = -\sum_{i=n+1}^{n+m} pt_i \cdot 2^{-i} \tag{2.24}$$

Since the truncated bits are m, $e_{\text{trunc}} \in \{-2^{-n} + 2^{-n-m}, 0\}$. $e_{\text{trunc}} = 0$ when all the discarded bits are equal to zero. $e_{\text{trunc}} = -2^{-n} + 2^{-n-m}$ when all the discarded bits are equal to 1 since $-\sum_{i=n+1}^{n+m} 1 \cdot 2^{-i} = -(2^{-n} - 2^{-n-m}) = -(\mathbf{lsb} - \mathbf{lsb_q})$.

The mean truncation error (μ_{trunc}) can be easily computed by assuming that each discarded bit pt_i has a probability 1/2 of being one:

$$\mu_{\text{trunc}} = -\sum_{i=n+1}^{n+m} \frac{1}{2} 2^{-i} = -\frac{\mathbf{lsb}}{2}\left(1 - 2^{-m}\right) \tag{2.25}$$

Similarly, assuming the bits pt_i independent and identically distributed, we can compute the variance σ^2_{trunc} of the truncation error:

$$\sigma^2_{\text{trunc}} = \sum_{i=n+1}^{n+m} \frac{1}{4} 2^{-2i} = \frac{\mathbf{lsb}^2}{12}\left(1 - 2^{-2m}\right) \tag{2.26}$$

2.2.2 Statistical Properties of the Erasing Error (e_{erasing})

The mean value of the erasing error is given by:

$$\mu_{\text{erasing}} = E\{e_{\text{erasing}}\} = \sum_{A\in\Theta} \left(\underset{IC=A}{E}\{e_{\text{erasing}}\} \right) P(A) \tag{2.27}$$

Since for each fixed IC $= A$, $f(\text{IC})$ is a constant value, using (2.19) the term within the summation in (2.27) can be written as:

$$\underset{IC=A}{E}\{e_{\text{erasing}}\} = f(A) - \underset{IC=A}{E}\{S_{\text{LSP}_{\text{minor}}}(x,y)\} \tag{2.28}$$

By using the definition (2.11) and by substituting (2.28) in (2.27) we have:

$$\mu_{\text{erasing}} = \sum_{A\in\Theta} [f(A) - \mu_{\text{LSP}}(A)] \cdot P(A) \tag{2.29}$$

Before computing the variance of the erasing error, let us calculate first the mean square erasing error, given by:

$$\varepsilon^2_{\text{erasing}} = E\{e^2_{\text{erasing}}\} \tag{2.30}$$

As done in the case of μ_{erasing}, we can compute $\varepsilon^2_{\text{erasing}}$ by performing the averaging on $\Omega(A)$, as follows:

$$\varepsilon^2_{\text{erasing}} = \sum_{A\in\Theta} \left(\underset{IC=A}{E}\{e^2_{\text{erasing}}\} \right) \cdot P(A) \tag{2.31}$$

where:

$$\underset{IC=A}{E}\{e^2_{\text{erasing}}\} = \underset{IC=A}{\text{VAR}}\{e_{\text{erasing}}\} + \left(\underset{IC=A}{E}\{e_{\text{erasing}}\} \right)^2 \tag{2.32}$$

Using (2.19), $\underset{IC=A}{\text{VAR}}\{e_{\text{erasing}}\} = \underset{IC=A}{\text{VAR}}\{f(A) - \mu_{\text{LSP}}(A)\}$, but for each fixed IC=A, $f(A)$ is a constant value as a consequence $\underset{IC=A}{\text{VAR}}\{e_{\text{erasing}}\} = \sigma^2_{\text{LSP}}$ (2.12). By using (2.28) one has:

$$\underset{IC=A}{E}\{e^2_{\text{erasing}}\} = \sigma^2_{\text{LSP}}(A) + (f(A) - \mu_{\text{LSP}}(A))^2 \tag{2.33}$$

By substituting (2.33) in (2.31) one finally obtains:

$$\varepsilon^2_{\text{erasing}} = \varepsilon^2_{\text{intrinsic}} + \varepsilon^2_{\text{comp}} \tag{2.34}$$

where:

$$\varepsilon^2_{\text{intrinsic}} = \sum_{A\in\Theta} \sigma^2_{\text{LSP}}(A)\cdot P(A) \tag{2.35}$$

$$\varepsilon^2_{\text{comp}} = \sum_{A\in\Theta} \left(f(A) - \mu_{\text{LSP}}(A)\right)^2 \cdot P(A) \tag{2.36}$$

If we define compensation error $e_{\text{comp}} = f(A) - \mu_{\text{LSP}}(A)$, $\varepsilon^2_{\text{comp}}$ is its mean square error. Furthermore the mean value and the variance of e_{comp} are given by:

$$\mu_{\text{comp}} = E\left\{f(A) - \mu_{\text{LSP}}(A)\right\} = \mu_{\text{erasing}} \tag{2.37}$$

$$\begin{aligned}\sigma^2_{\text{comp}} &= \sum_{A\in\Theta} \left(f(A) - \mu_{\text{LSP}}(A) - \mu_{\text{comp}}\right)^2 \cdot P(A) = \\ &= \sum_{A\in\Theta} \left(f(A) - \mu_{\text{LSP}}(A) - \mu_{\text{erasing}}\right)^2 \cdot P(A)\end{aligned} \tag{2.38}$$

As $\sigma^2_{\text{erasing}} = \varepsilon^2_{\text{erasing}} - \mu^2_{\text{erasing}}$, using (2.34), (2.37), the variance of the erasing error ($\sigma^2_{\text{erasing}}$) is given by:

$$\sigma^2_{\text{erasing}} = \varepsilon^2_{\text{intrinsic}} + \sigma^2_{\text{comp}} \tag{2.39}$$

2.2.3 Optimal Compensation Function and Error Lower Bound

From (2.20),(2.21),(2.23),(2.39) one has:

$$\begin{aligned}\varepsilon^2_{\text{total}} \simeq\ & \sigma^2_{\text{trunc}} + \varepsilon^2_{\text{intrinsic}} + \sigma^2_{\text{comp}} \\ & + \left(\mu_{\text{erasing}} + \mu_{\text{trunc}} + K_{\text{round}}\right)^2\end{aligned} \tag{2.40}$$

This equation highlights that the rounding constant K_{round} should be chosen in order to minimize μ_{total}. In the ideal case in which K_{round} is represented on an infinite number of bits (i.e., $m \to \infty$ in the scheme of Fig. 2.1 and $f(\text{IC})$ and K_{round} are real numbers) we can choose:

$$K_{\text{round}} = K_{\text{round}_{\text{ideal}}} = -\mu_{\text{erasing}} - \mu_{\text{trunc}} \tag{2.41}$$

and we can compensate the mean values of e_{erasing} and e_{trunc} exactly. In practice, $f(\text{IC})$ and K_{round} will be expressed on a finite number of bits (see Fig. 2.1) and the mean value of e_{total} can not be exactly zero. We will discuss this aspect in Sec. 2.5, when we will discuss the effect of quantization. For now let us observe that in the ideal case in which the compensation function

$f(\mathrm{IC})$ (and K_{round}) returns a real number, μ_{total} can be nullified by choosing K_{round} following (2.41), and we have:

$$\varepsilon^2_{\mathrm{total}} \simeq \sigma^2_{\mathrm{trunc}} + \varepsilon^2_{\mathrm{intrinsic}} + \sigma^2_{\mathrm{comp}} \tag{2.42}$$

This equation shows that the mean square error of the truncated multiplier can be reduced to the sum of three positive terms.

The first term, $\sigma^2_{\mathrm{trunc}}$, is the variance of truncation error. This error component arises from using only n output bits from the multiplier. This error component is (almost) independent from the technique used to realize the truncated multiplier and is also presented in the full-rounded multiplier (see Fig. 1.11). In the ideal case in which $f(\mathrm{IC})$ is real ($m \to \infty$), from (2.26), $\sigma^2_{\mathrm{trunc}}$ is given by $\mathbf{lsb}/12 \simeq 0.083 \cdot \mathbf{lsb}^2$.

The second term, $\varepsilon^2_{\mathrm{intrinsic}}$, is not dependent on the compensation function. This error component arises since we use only the elements in IC to estimate $S_{\mathrm{LSP_{minor}}}$.

The third term, σ^2_{comp}, depends from the compensation function and can be minimized with a suitable selection of $f(\mathrm{IC})$. It is worth noting that σ^2_{comp} does not change if we add a constant to the function $f(\mathrm{IC})$. As a consequence, there are infinite optimal compensation functions differing one another from a constant value. However, it is easily found from the above discussion that there is only one optimal value for the sum: $f(\mathrm{IC}) + K_{\mathrm{round}}$.

From (2.38) and (2.29) it can be seen that an optimal choice for the compensation function is:

$$f_{opt}(\mathrm{IC}) = \mu_{\mathrm{LSP}} \tag{2.43}$$

In fact, when (2.43) is verified, all the terms summed in (2.38) are zero and hence $\sigma^2_{\mathrm{comp}} \simeq 0$. Thus, an optimal compensation function is the one that returns the mean value of $S_{\mathrm{LSP_{minor}}}$ for every value of the IC. The compensation function in (2.43) is the one that makes equal to zero the mean erasing error (μ_{erasing}). From (2.41), the corresponding optimal rounding constant is $K_{\mathrm{round\_ideal}} = -\mu_{\mathrm{trunc}}$. The mean square error when the optimal compensation function and the optimal rounding constant are employed is a lower error bound for any variable-correction truncated multiplier and is given by:

$$\varepsilon^2_{\mathrm{low\_bound}} \simeq \sigma^2_{\mathrm{trunc}} + \varepsilon^2_{\mathrm{intrinsic}} \tag{2.44}$$

where, as previously discussed, $\sigma^2_{\mathrm{trunc}} \simeq \mathbf{lsb}/12$.

2.3 Optimal Compensation Function

The generic element $x_i y_j$ of $\text{LSP}_{\text{minor}}$ is correlated to two elements of IC: $\gamma_{i-h} = x_i y_{n+1+h-i}$ and $\gamma_{n+1-j} = x_{n+1+h-j} y_j$. For example in Fig. 2.1 $x_6 y_7$ is correlated to $\gamma_4 = x_6 y_5$ and $\gamma_2 = x_4 y_7$. In the hypotheses of input bits independent and identically distributed, as $x_i y_j$ is correlated to different elements of the IC, its mean value is given by:

$$\underset{IC=A}{E}\{x_i y_j\} = \underset{IC=A}{E}\{x_i\} \cdot \underset{IC=A}{E}\{y_j\} \tag{2.45}$$

The generic $\gamma_{i-h} = x_i y_{n+1+h-i}$ is given by an *AND* operation, so if $\gamma_{i-h} = 1$, x_i will be 1 with probability equal to 1, otherwise with a probability equal to $1/3$ (one of the three left cases):

$$\underset{IC=A}{E}\{x_i\} = \frac{1}{3}\left(1 + 2\gamma_{i-h}\right) \tag{2.46}$$

Analogous reasoning is valid for the other element of partial product. Hence the mean value of a generic element of $\text{LSP}_{\text{minor}}$ can be expressed as:

$$\mu_{i,j}(\text{IC}) \triangleq \underset{IC=A}{E}\{x_i y_j\} = \frac{1}{9}\left(1 + 2\gamma_{i-h}\right)\left(1 + 2\gamma_{n+1-j}\right) \tag{2.47}$$

$$\mu_{i,j}(\text{IC}) \in \left\{\frac{1}{9}, \frac{1}{3}, 1\right\}$$

In order to express μ_{LSP} in explicit form, let us start by computing the mean values of $\text{LSP}_{\text{minor}}$ when all the elements of the IC are zero. In this case, using 2.47, the mean value of each partial product of $\text{LSP}_{\text{minor}}$ is $1/9$. It can be easily seen also if we consider all the possible values of the input bits, constrained by the condition $\text{IC} = [0,0,0,0,0,0]$. For example, Fig. 2.2(a) shows the computation of the mean value of the partial product $x_6 y_7$, whose value results to be $1/9$. Therefore, if we replace each partial product with its mean (Fig. 2.2(b)), with simple algebra one obtains:

$$\mu_{\text{LSP}}\left(\text{IC} = 0\right) = \mathbf{lsb} \cdot 2^{-h-1} \cdot K \tag{2.48}$$

where

$$K = \sum_{i=h+2}^{n} \sum_{j=n+h+2-i}^{n} \frac{1}{9} 2^{-i-j} = \frac{2}{9}\left(\frac{n_{eq}}{2} + 2^{-n_{eq}} - 1\right) \tag{2.49}$$

Next step is to consider the case in which only one element of the IC is equal to 1. When a generic $\gamma_i = x_{h+i} y_{n+1-i}$ is equal to 1, x_{h+i} and y_{n+1-i}

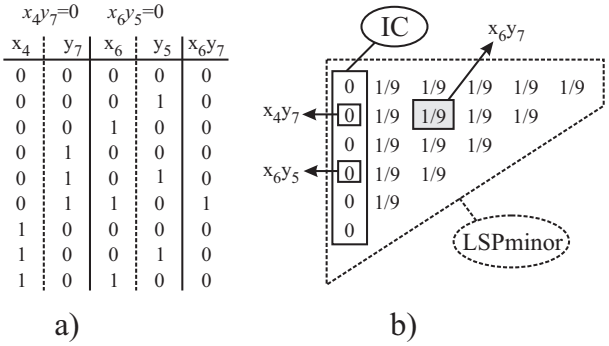

Figure 2.2: Computation of μ_{LSP} for $n = 8$ unsigned multiplier with IC $= 0$. a) Computation of the mean value of x_6y_7. b) Mean value of the elements of $\text{LSP}_{\text{minor}}$.

are both equal to 1. This change the probability of being 1 of the elements of $\text{LSP}_{\text{minor}}$ correlated to them, that is the elements on the same diagonal $x_{h+i}y_j \;\; (j = n+2-i, \ldots, n)$ and on the same row $x_j y_{n+1-i} \;\; (j = h+i+1, \ldots, n)$ (on the grey in Fig. 2.3(a)). Using (2.47) the mean value of these elements is $1/3$. Thus for each γ_i equal to 1, the value of μ_{LSP} can be obtained by adding to (2.48) a correction factor which is equal to:

$$f_i = 2^{-n-h-1}\left(1 + \sum_{j=n+2-i}^{n} \frac{2}{9}2^{n+1-i-j} + \sum_{j=h+i+1}^{n} \frac{2}{9}2^{-i}\right) \tag{2.50}$$

in which the first element consider the increment in IC column due to the $\gamma_i = 1$, the second term the increment to (2.48) due to the change in the elements on the same diagonal, the third the change due to the elements on the same row. Considering all the possible cases in which only one element of the IC ($\gamma_i, i = 1 \ldots n_{eq}$) is equal to 1 yields:

$$\mu_{\text{LSP}}|_{\text{only one } \gamma_i=1} = \mu_{\text{LSP}}\,(\text{IC} = 0) + \mathbf{lsb} \cdot 2^{-h-1} \sum_{i=1}^{n_{eq}} f_i\gamma_i \tag{2.51}$$

where:

$$f_i = \frac{13}{9} - \frac{2}{9}\left(2^{-i+1} + 2^{i-n_{eq}}\right). \tag{2.52}$$

The above reasoning can be iterated to compute the value of μ_{LSP} when two or more elements of the IC are equal to 1. In Fig. 2.3(b) two elements x_4y_7 and

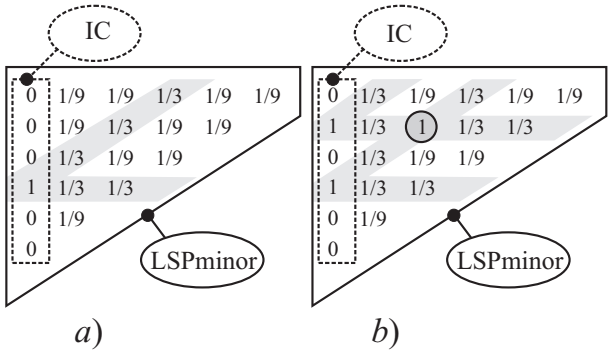

Figure 2.3: Mean values of the elements of the LSP when IC is non-zero for a 8×8 multiplier ($h = 2$). a) Only one element of the IC is equal to 1. b) Two or more elements of the IC are equal to 1.

x_6y_5 of the IC are equal to 1. In this case the value of μ_{LSP} differs from (2.51) due to the mean value of the partial product x_6y_7 (circled in Fig. 2.3(b)). In fact, since $x_4 = y_7 = x_6 = y_5 = 1$, the mean value of x_6y_7 is 1, whereas in (2.51) it is computed as $\left(\frac{1}{9} + \frac{2}{9} + \frac{2}{9}\right)$. Hence the mean value of x_6y_7 should be increased by a factor $\frac{4}{9}$ with respect to (2.51). In the general case, when $\gamma_i = x_{h+i}y_{n+1-i}$ and $\gamma_j = x_{h+j}y_{n+1-j}$ are both equal to 1, for the common element $x_{h+j}y_{n+1-i}$ $(i < j)$ it should be considered an increment of:

$$f_{i,j} = \frac{4}{9}2^{-\text{abs}(j-i)} \tag{2.53}$$

Finally $f_{opt}(\text{IC}) = \mu_{\text{LSP}}(\text{IC})$ can be computed as:

$$f_{opt}(\text{IC}) = \mathbf{lsb} \cdot 2^{-h-1}\left[K + \sum_{i=1}^{n_{eq}} f_i\gamma_i + \sum_{i=1}^{n_{eq}}\sum_{j=i+1}^{n_{eq}} f_{i,j}\gamma_i\gamma_j\right] \tag{2.54}$$

where K, f_i and $f_{i,j}$ are defined in (2.49), (2.52) and (2.53).

Discussion

The optimal compensation function in (2.54) is a quadratic form in the variables γ_i. The value of (2.54) decreases exponentially with h, as expected, since larger h means lower weight of the terms in $\text{LSP}_{\text{minor}}$.

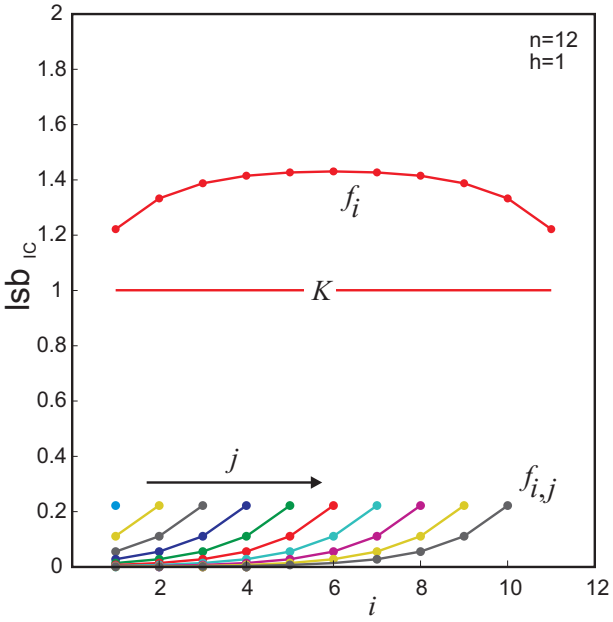

Figure 2.4: Coefficients of the optimal error compensation function $f_{opt}(\mathrm{IC})$ for $n = 12, h = 1$.

Fig. 2.4 shows the behavior of coefficients K, f_i and $f_{i,j}$ in (2.54) for a truncated multiplier with $n = 12$ and $h = 1$. The linear terms f_i exhibit a maximum for $i = \frac{1}{2}(1 + n_{eq})$ and the maximum f_i value is: $f_{i,\max} = \frac{13}{9} - \frac{4}{9} \cdot 2^{\frac{1}{2}(1-n_{eq})}$. The value of $f_{i,\max}$ tends to $\frac{13}{9}$ as n_{eq} increases. As shown in Fig. 2.4, the maximum in the distribution of f_i is not very pronounced. The minimum of the f_i is reached for both $i = 1$ and $i = n_{eq}$, is given by: $f_{i,\min} = \frac{11}{9} - \frac{4}{9}\, 2^{-n_{eq}}$ and tends to $\frac{11}{9}$ as n_{eq} increases.

The quadratic terms $f_{i,j}$ decrease exponentially with $\mathrm{abs}(j-i)$. The maximum of the quadratic coefficients, attained for $\mathrm{abs}(j-i) = 1$, is 2/9. Thus, the quadratic coefficients are significantly smaller than the linear terms.

The constant term K in (2.49) increases almost linearly with n_{eq}. In practice, K is of the same order of magnitude as f_i (in the example of Fig. 2.4 K is actually smaller than $f_{i,\min}$). Therefore, the optimal compensation function (2.54) is far from being constant. This explains the rather large approximation

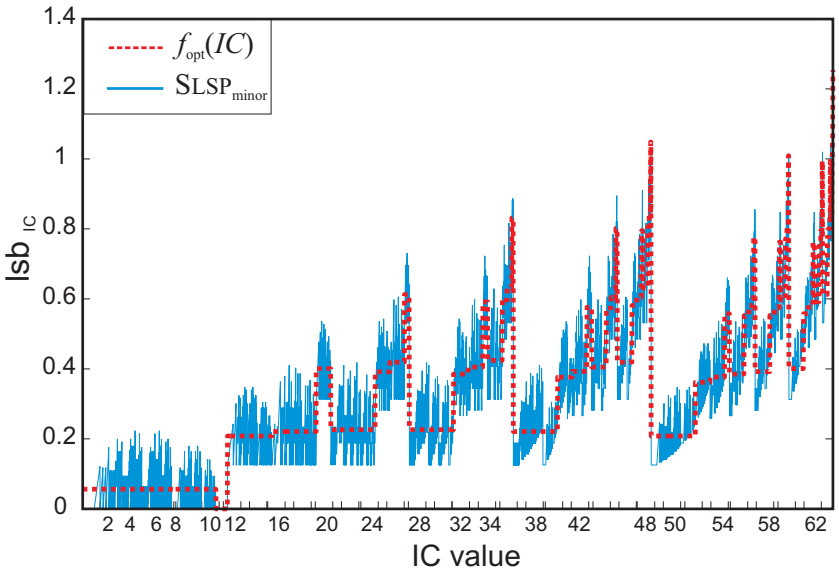

Figure 2.5: Comparison between the optimal error compensation function $f_{opt}(\text{IC})$, red dashed line, and the $S_{\text{LSP}_{\text{minor}}}$, green full line, for all the possible input values ($n = 6, h = 0$, collected according to the IC value.

error of the constant-correction techniques [8, 9, 11].

In Fig. 2.5 there is the comparison between the value of $f_{opt}(\text{IC})$ and the $S_{\text{LSP}_{\text{minor}}}$, for all the possible input values ($n = 6$, $h = 0$), collected according to the IC value. It is clear that for each IC value $f_{opt}(\text{IC})$ is the mean of all the possible values assumed by $S_{\text{LSP}_{\text{minor}}}$.

2.3.1 The Intrinsic Error

The intrinsic error is given by (2.35). This equation shows that the intrinsic error is a weighted sum of the variances $\sigma^2_{\text{LSP}}(\text{A})$ of $S_{\text{LSP}_{\text{minor}}}$ in the set $\Omega(A)$. From (2.10), in turn, $S_{\text{LSP}_{\text{minor}}}$ is a weighted sum of partial products. For each fixed IC $= A$ the summation of the elements of the IC is a constant value, so they will not appear in the calculation of the variance of $S_{\text{LSP}_{\text{minor}}}(\text{A})$. Hence, $\sigma^2_{\text{LSP}}(\text{A})$ can be computed as the sum of the variances of each partial product $\sigma^2_{i,j}(A)$ (multiplied by the square of partial product weight) summed

to twice the covariance between every couple of different partial products $\mathrm{COV}_{i,j,l,m}(A)$ (multiplied by the products of their weights) taken once. Since the variables are binary numbers $(x_i y_j)^2 = x_i y_j$ and the variances are computed as:

$$\begin{aligned}\sigma^2_{i,j}(A) &= \underset{IC=A}{E}\left\{(x_i y_j)^2\right\} - \mu^2_{i,j}(A) = \\ &= \underset{IC=A}{E}\{x_i y_j\} - \mu^2_{i,j}(A) = \mu_{i,j}(A) - \mu^2_{i,j}(A)\end{aligned} \tag{2.55}$$

where $\mu_{i,j}$ is given by (2.47).

The covariance terms are given by:

$$\mathrm{COV}_{i,j,l,m}(A) = \underset{IC=A}{E}\{x_i y_j \cdot x_l y_m\} - \mu_{i,j}(A)\mu_{l,m}(A) \tag{2.56}$$

App. A describes how to compute the mean of the product $x_i y_j \cdot x_l y_m$ in every set $\Omega(A)$ and the complete calculation of the explicit form of the intrinsic error (2.35). The final expression is given by:

$$\begin{aligned}\varepsilon^2_{\text{intrinsic}} = \mathbf{lsb}^2 \cdot 2^{-2h}\Bigg[&\frac{1}{24}2^{-n_{eq}} - \frac{7}{324}2^{-2n_{eq}} + \\ &+ \left(\frac{13}{864} - \frac{1}{48}2^{-n_{eq}}\right)\cdot n_{eq} - \frac{13}{648}\Bigg]\end{aligned} \tag{2.57}$$

The above exact equation can be simplified by neglecting the small terms depending on $2^{-n_{eq}}$:

$$\varepsilon^2_{\text{intrinsic}} \simeq \mathbf{lsb}^2 \cdot 2^{-2h}\frac{13}{216}\left[\frac{1}{4}\cdot n_{eq} - \frac{1}{3}\right] \tag{2.58}$$

The relative error of (2.58) with respect to (2.57) is less than 4% for $n_{eq} = 5$ and rapidly vanishes for larger n_{eq} values, as it is possible to see in Fig. 2.6. In the figure the red full line is the value of the exact expression of the intrinsic error (2.57), the dotted black line the value of its approximation (2.58). The two lines are almost indistinguishable, a small difference is visible only for small values.

In the specific case in which $h = 0$ the intrinsic error is given by:

$$\varepsilon^2_{\text{intrinsic}} \simeq \mathbf{lsb}^2 \cdot [0.015 \cdot n - 0.02] \tag{2.59}$$

Hence the intrinsic error increases linearly with *n*, when *h*=0.

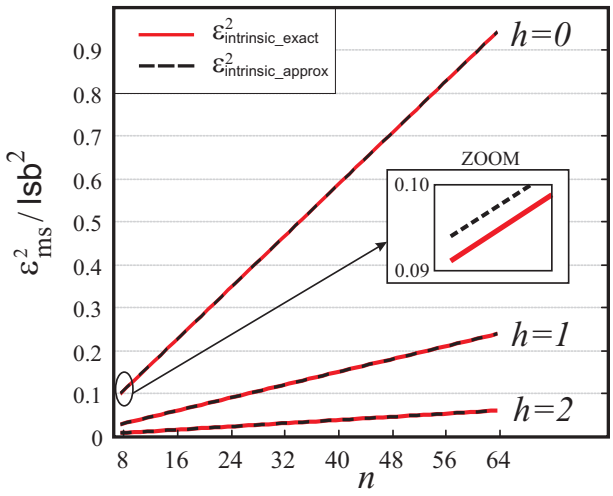

Figure 2.6: Comparison between the exact value of $\varepsilon^2_{\text{intrinsic}}$ (red full line) and the approximate value of $\varepsilon^2_{\text{intrinsic}}$ (black dashed line).

In general, for each given h value the intrinsic error increases linearly with n (with $n_{eq} = n - h$), while for each given n value the intrinsic error decreases exponentially with h, going to zero quickly (see Fig. 2.6).

The total mean square error introduced by the optimal compensation function, $\varepsilon^2_{\text{low\_bound}}$, can be obtained in closed form using (2.57) in (2.44). Its value, varying n and h, is shown in Fig. 2.7 (red full lines) compared to the error of the full-rounded multiplier (green dotted line). The figure shows that the lower bound for any truncated multiplier is far from $\varepsilon^2_{\text{round}} = 0.083 \cdot \mathbf{lsb}^2$, especially for high values of n and low values of h. This is an important point. Since now it has been thought that using a suitable compensation function, the lowest error that could be obtained was $\varepsilon^2_{\text{round}}$; but it is impossible since it has been demonstrated that the lower error is represented by $\varepsilon^2_{\text{low\_bound}} > \varepsilon^2_{\text{round}}$ due to the presence of $\varepsilon^2_{\text{intrinsic}}$ which is independent by $f(\text{IC})$. For example if a designer want to implement a 16 bit truncated multiplier with a mean square error lower than $0.2 \cdot \mathbf{lsb}^2$, the only things that he can do is to look for a compensation function of a truncated multiplier with $h = 1$.

Fig. 2.7 shows that the error increases almost linearly with n, following the slope of $\varepsilon^2_{\text{intrinsic}}$. When $h = 0$ the overall error is much larger $\varepsilon^2_{\text{round}} = 0.083 \cdot \mathbf{lsb}^2$. On the other hand since, from (2.58), the intrinsic error decreases

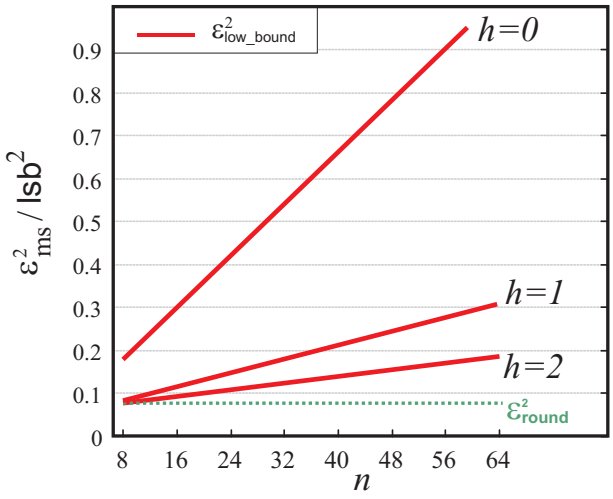

Figure 2.7: $\varepsilon^2_{\text{low\_bound}}$ values varying n and h, compared with $\varepsilon^2_{\text{round}}$.

exponentially with h, Fig. 2.7 shows that the overall error approaches $\varepsilon^2_{\text{round}}$ as h increases.

2.4 Linear compensation function.

The optimal compensation function obtained in Sec. 2.3, while being optimal in terms of mean square error, can hardly be implemented in hardware. The equation (2.54) requires in fact the sum of a large number of terms, in the order of n^2_{eq}. Furthermore the coefficients K, f_i and $f_{i,j}$ in (2.54) should be quantized with a few bits of precision, to keep the area and power advantages of truncated multipliers.

As observed before, the quadratic coefficients $f_{i,j}$ are significantly smaller than the linear terms. This suggests that a good approximation of the best possible error compensation function can be obtained as a simple linear combination of the elements of the IC:

$$f_{lin}(\text{IC}) = \mathbf{lsb} \cdot 2^{-h-1} \left[K_l + \sum_{i=1}^{n_{eq}} l_i \gamma_i \right] \tag{2.60}$$

The use of the sub-optimal $f_{lin}(\text{IC})$ in place of $f_{opt}(\text{IC})$ will increase the mean square error. On the other hand the implementation of $f_{lin}(\text{IC})$ requires only

n_{eq} sums, thus reducing the hardware complexity of the multiplier. Furthermore, as we will show in the following, the coefficients in (2.60) can be represented on just one or two bits with a limited error increase.

The expression of the total mean square error from $\varepsilon^2_{\text{low\_bound}} = \sigma^2_{\text{trunc}} + \varepsilon^2_{\text{intrinsic}}$ becomes $\varepsilon^2_{\text{low\_bound}} = \sigma^2_{\text{trunc}} + \varepsilon^2_{\text{intrinsic}} + \sigma^2_{\text{comp}}$. There is a component $\sigma^2_{\text{comp}} \neq 0$ which arises from using a linear approximation of $f_{opt}(\text{IC})$ (hence so the compensation error $e_{\text{comp}} = f(\text{IC}) - \mu_{\text{LSP}} = f(\text{IC}) - f_{opt}(\text{IC})$ is not equal to zero). The optimal values of the coefficients K_l, l_i in (2.60) can be obtained by minimizing this new component $\sigma^2_{\text{lin}} = \sigma^2_{\text{comp}}|_{f(\text{IC})=f_{lin}(\text{IC})}$ (2.38), given by:

$$\sigma^2_{\text{lin}} = \sum_{A\in\Theta} \left(f_{lin}(A) - \mu_{\text{LSP}}(A) - \mu_{\text{lin}}\right)^2 P(A) \tag{2.61}$$

Where $f_{lin}(\text{IC})$ is given by (2.60), $\mu_{\text{LSP}}(A) = f_{opt}(\text{IC})$ by (2.54) and μ_{lin} is given by (2.29) with $f(\text{IC}) = f_{lin}(\text{IC})$. Since K_l is present both in $f_{lin}(\text{IC})$ and in μ_{lin}, σ^2_{lin} doesn't depend on the constant K_l (see also App. B). Therefore the optimal l_i coefficients are obtained by solving the following system:

$$\begin{cases} \dfrac{\partial \sigma^2_{\text{lin}}}{\partial l_m} = 0 \quad m = 1, ..., n_{eq} \end{cases} \tag{2.62}$$

This is a linear system of n_{eq} equations in the n_{eq} unknowns l_i, which solution is detailed in the App. B. The result is:

$$l_i = \frac{5}{3} - \frac{1}{3}\left(2^{-i+1} + 2^{i-n_{eq}}\right) \tag{2.63}$$

The value of K_l is chosen imposing $\mu_{\text{lin}} = 0$, that is:

$$\mu_{\text{lin}} = \sum_{A\in\Theta} \left(f_{lin}(A) - \mu_{\text{LSP}}(A)\right) \cdot P(A) = 0 \tag{2.64}$$

Note that by imposing $\mu_{\text{lin}} = 0$, from (2.61) and (2.36), we minimize not only σ^2_{comp} but also $\varepsilon^2_{\text{comp}}$. The solution of (2.64) is again detailed in the App. B. The result is:

$$K_l = \frac{1}{6}\left(\frac{n_{eq}}{2} + 2^{-n_{eq}} - 1\right) \tag{2.65}$$

Fig. 2.8 shows the behavior of coefficients K_l and l_i for a truncated multiplier with $n = 12$ and $h = 1$. Also in this case, as for Fig. 2.8, the distribution of linear terms l_i is symmetric and exhibits a maximum for $i = (1 + n_{eq})/2$.

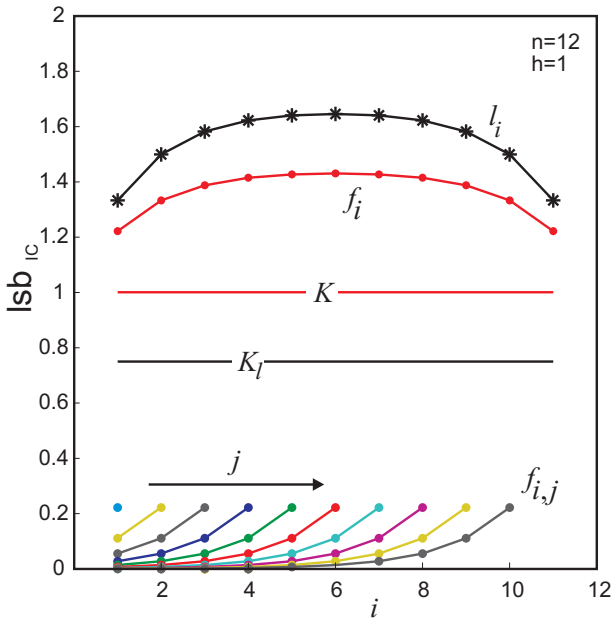

Figure 2.8: Comparison between the coefficients of the linear compensation function $f_{lin}(\text{IC})$, black line with asterisk, and of the optimum compensation function $f_{opt}(\text{IC})$, red line with circle, for $n = 12, h = 1$.

The maximum l_i value is: $l_{i,\max} = \frac{5}{3} - \frac{1}{3} \cdot 2^{\frac{1}{2}(1-n_{eq})}$ and tends to 5/3 as n_{eq} increases. The minimum of the l_i is reached for both $i = 1$ and $i = n_{eq}$, is given by: $l_{i,\min} = \frac{4}{3} - \frac{2}{3}\, 2^{-n_{eq}}$ and tends to 4/3 as n_{eq} increases. In the figure are also shown coefficients and the constant of the optimum compensation function $f_{opt}(\text{IC})$ for the comparison. Note that $l_i > f_i$ since they have to compensate the value of f_{ij} erased in linear function. The value of the constant K_l is lower than K but it's worth noticing that it should be add a suitable rounding constant.

Fig. 2.9 shows the comparison between the value of $f_{opt}(\text{IC})$ and the $f_{lin}(\text{IC})$, for all the possible input values ($n = 6$, $h = 0$), collected according to the IC value. As it is shown in the figure $f_{lin}(\text{IC})$ is a good approximation of $f_{opt}(\text{IC})$.

The value of σ^2_{lin} obtained with the coefficients (2.63) can be calculated by

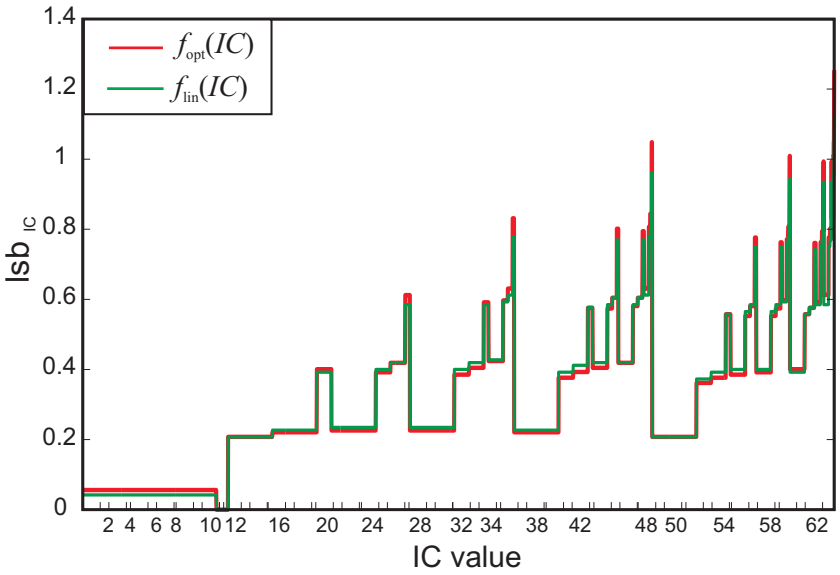

Figure 2.9: Comparison between the optimal error compensation function and the linear compensation function, for all the possible input values ($n = 6$, $h = 0$), collected according to the IC value. Red full line: $f_{opt}(\mathrm{IC})$. Green full line: $f_{lin}(\mathrm{IC})$.

substituting (2.60) and (2.63) in (2.61). After some tedious algebraic manipulation, one has:

$$\sigma^2_{\mathrm{lin}} = \mathbf{lsb}^2 \cdot \frac{1}{1296} \cdot 2^{-2h} \left[\frac{3}{4} \cdot n_{eq} + 2^{-2n_{eq}} - 1 \right] \tag{2.66}$$

The above exact equation can be simplified by neglecting the small term in $2^{-2n_{eq}}$:

$$\sigma^2_{\mathrm{lin}} \simeq \mathbf{lsb}^2 \cdot \frac{3}{1296} \cdot 2^{-2h} \left[\frac{1}{4} \cdot n_{eq} - \frac{1}{3} \right] \tag{2.67}$$

The overall error obtained with the optimal linear compensation function can be written as:

$$\varepsilon^2_{\mathrm{total\_lin}} \simeq \sigma^2_{\mathrm{trunc}} + \varepsilon^2_{\mathrm{intrinsic}} + \sigma^2_{\mathrm{lin}} \tag{2.68}$$

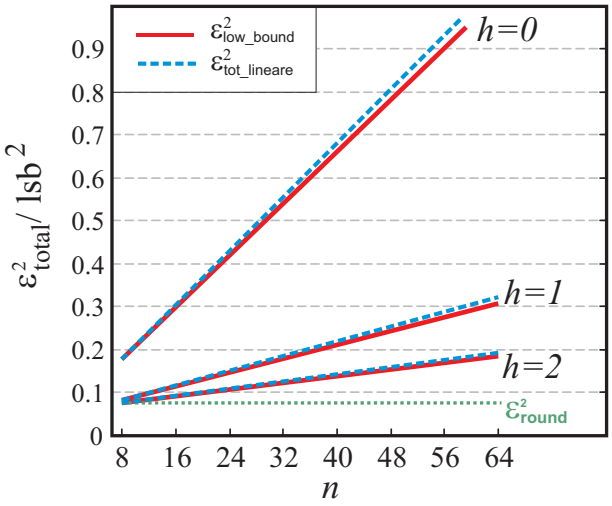

Figure 2.10: Normalized total mean square error as a function of n and h. Red full line: error of optimal (quadratic) compensation function. Blue dashed line: linear compensation function with optimal coefficient. Green dotted line: mean square error of the full-rounded multiplier.

By comparing (2.67) and (2.58) it comes out that, since $n_{eq} \gg 4/3$:

$$\frac{\sigma^2_{\text{lin}}}{\varepsilon^2_{\text{intrinsic}}} \simeq \frac{1}{26} \simeq 0.038 \tag{2.69}$$

Therefore, the use of the linear compensation function (2.60), in place of the optimal function (2.54), introduces only a small increase in the mean square error, lower than 3.8%. This is also shown in Fig. 2.10, where the total error achieved by using the linear compensation function $\varepsilon^2_{tot\_linear}$ is reported as a blue line and compared with the $\varepsilon^2_{\text{low\_bound}}$.

2.5 Linear Coefficients Quantization

In order to implement the linear compensation function efficiently, the coefficients given by (2.63),(2.65) and the rounding constant K_{round} should be

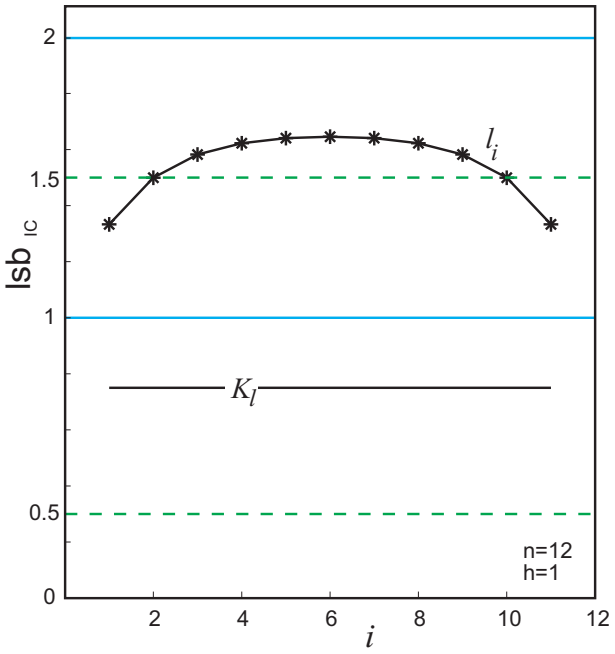

Figure 2.11: Levels of quantization for l_i. When $\mathbf{lsb_q} = \mathbf{lsb_{IC}}$, only the blue full lines are admitted. When $\mathbf{lsb_q} = 1/2\mathbf{lsb_{IC}}$, also the green dashed levels can be used.

quantized on a reduced number of bits. Coefficient quantization will introduce other error components, in addition to the terms reported in (2.68). As shown in Fig. 2.1, both $f(\text{IC})$ and the rounding constant K_{round} are represented on $n + m$ bits, that is with a less significant bit equal to $\mathbf{lsb_q}$ (2.4). Clearly, a larger m leads to a lower error due to coefficient quantization, but the hardware complexity increases. In the following we will consider two cases: $\mathbf{lsb_q} = \mathbf{lsb_{IC}}$ and $\mathbf{lsb_q} = \frac{1}{2}\mathbf{lsb_{IC}}$. In the next chapter we will discuss efficient hardware implementations for both cases. The quantization level are represented in Fig. 2.11. When $\mathbf{lsb_q} = \mathbf{lsb_{IC}}$ the q_i are integer values (blue full line), whereas the q_i are multiple of 1/2 for $\mathbf{lsb_q} = \frac{1}{2}\mathbf{lsb_{IC}}$, hence it is possible to consider also the green dashed line.

The quantized-coefficients error compensation function $f_q(\text{IC})$ can be ex-

pressed as:

$$f_q(\mathrm{IC}) = \mathbf{lsb_{IC}} \cdot \left[\sum_{i=1}^{n_{eq}} q_i \gamma_i \right] \tag{2.70}$$

where:

$$q_i = l_i + \Delta l_i \tag{2.71}$$

are the quantized coefficients and Δl_i (i=1,...,n_{eq}) represent the effect of coefficient quantization. Please note that no constant is added in (2.70). As already discussed in Sec. 2.2.3, what actually matters is the sum of $f(\mathrm{IC})$ and the rounding constant. Thus, the effect of the constant K_l in (2.60) will be taken into account when computing the rounding constant K_{round}.

In order to determine the effect of the quantization on the multiplier error, let us call $\Delta f_{lin}(\mathrm{IC})$ the difference between $f_q(\mathrm{IC})$ and $f_{lin}(\mathrm{IC})$:

$$\Delta f_{lin}(\mathrm{IC}) = LSB_{IC} \cdot \left(\sum_{i=1}^{n_{eq}} \Delta l_i \gamma_i - K_l \right) \tag{2.72}$$

Since $f_q(IC) = f_{lin}(\mathrm{IC}) + \Delta f_{lin}(\mathrm{IC})$, the mean erasing error after coefficients quantization, μ_{quant}, can be easily obtained from (2.29) as:

$$\begin{aligned} \mu_{\mathrm{quant}} = & \sum_{A \in \Theta} \left[f_{lin}(A) - \mu_{\mathrm{LSP}}(A) \right] \cdot P(A) + \\ & + \sum_{A \in \Theta} \Delta f_{lin}(A) \cdot P(A) \end{aligned} \tag{2.73}$$

The first term in (2.73) is zero (see (2.64)) and therefore μ_{quant} can be written as:

$$\mu_{\mathrm{quant}} = \sum_{A \in \Theta} \Delta f_{lin}(A) \cdot P(A) = E\left\{ \Delta f_{lin} \right\} \tag{2.74}$$

By using (2.72) and by observing that the mean value of γ_i is 1/4, one obtains:

$$\begin{aligned} \mu_{\mathrm{quant}} &= \sum_{A \in \Theta} \mathbf{lsb_{IC}} \cdot \left(\sum_{i=1}^{n_{eq}} \Delta l_i \gamma_i - K_l \right) \cdot P(A) \\ &= \mathbf{lsb_{IC}} \left(\frac{1}{4} \cdot \sum_{i=1}^{n_{eq}} \Delta l_i - K_l \right) \end{aligned} \tag{2.75}$$

In order to compute σ^2_{comp} after coefficient quantization, $\sigma^2_{\mathrm{comp\_quant}}$, we can start from (2.38) and, after a few algebraic manipulation, we obtain:

$$\sigma^2_{\mathrm{comp\_quant}} = \sigma^2_{\mathrm{lin}} + \sigma^2_{\mathrm{q}} \tag{2.76}$$

where:

$$\sigma_{\mathrm{q}}^2 = \sum_{A\in\Theta} \left(\Delta f_{lin}(A) - \mu_{\mathrm{quant}}\right)^2 \cdot P(A) = \\ = E\left\{\left(\Delta f_{lin}(\mathrm{IC}) - \mu_{\mathrm{quant}}\right)^2\right\} = VAR\left\{\Delta f_{lin}(\mathrm{IC})\right\} \tag{2.77}$$

where the variance of $\Delta f_{lin}(\mathrm{IC})$ can be easily computed by using (2.72):

$$\sigma_{\mathrm{q}}^2 = \mathbf{lsb}_{\mathbf{IC}}^2 \frac{3}{16} \cdot \sum_{i=1}^{n_{eq}} \Delta l_i^2 \tag{2.78}$$

2.5.1 Optimal Quantized Coefficients

Let us consider the total multiplier error, given by (2.40). By using (2.76) we have:

$$\varepsilon_{\mathrm{total}}^2 \simeq \sigma_{\mathrm{trunc}}^2 + \varepsilon_{\mathrm{intrinsic}}^2 + \sigma_{\mathrm{lin}}^2 + \sigma_{\mathrm{q}}^2 \\ + \left(\mu_{\mathrm{quant}} + \mu_{\mathrm{trunc}} + K_{\mathrm{round}}\right)^2 \tag{2.79}$$

In this equation the coefficient quantization influences the fourth term σ_{q}^2 (see (2.78)), and the last term. In fact, K_{round} is quantized and cannot assume the ideal value $K_{round\_ideal} = -\mu_{\mathrm{quant}} - \mu_{\mathrm{trunc}}$ (2.41) exactly.

The problem of finding the optimal quantized coefficients q_i (that is the Δl_i values) and the optimal quantized constant K_{round} can be written as follows:

$$\sigma_{\mathrm{q}}^2 + \left(\mu_{\mathrm{quant}} + \mu_{\mathrm{trunc}} + K_{\mathrm{round}}\right)^2 \quad \text{min!} \tag{2.80}$$

where "min!" is used in order to indicate that the quantity should be minimized. By using (2.78) and (2.75) the optimization problem can be reformulated as:

$$\frac{3}{16}\sum_{i=1}^{n_{eq}} \Delta l_i^2 + \left(\frac{K_{\mathrm{round}} + \mu_{\mathrm{trunc}}}{\mathbf{lsb}_{\mathbf{IC}}} + \frac{1}{4}\cdot\sum_{i=1}^{n_{eq}} \Delta l_i - K_l\right)^2 \quad \text{min!} \tag{2.81}$$

Since Δl_i and K_{round} are quantized, (2.81) is an integer quadratic problem.

In order to simplify this problem we can observe that K_{round} appears only in the second term of (2.81). Therefore, for given q_i (and hence Δl_i) values, the best choice of K_{round} is the one that minimizes the second term of (2.81). By using (2.75) and (2.25) one obtains:

$$K_{\mathrm{round}} = \frac{\mathbf{lsb}}{2} + \mathrm{round}_{\mathbf{lsb_q}}\left[\mathbf{lsb}_{\mathbf{IC}}\left(K_l - \frac{1}{4}\sum_{i=1}^{n_{eq}} \Delta l_i\right) - \frac{\mathbf{lsb_q}}{2}\right] \tag{2.82}$$

where we have exploited the fact that $\mathbf{lsb}/(2\mathbf{lsb_q})$ is integer. The Equation (2.82) gives in closed form the optimal rounding constant K_{round}, given the quantized coefficients q_i.

The analytical solutions of the integer quadratic optimization problem (2.81) in the two cases $\mathbf{lsb_q} = \mathbf{lsb_{IC}}$ and $\mathbf{lsb_q} = \frac{1}{2}\mathbf{lsb_{IC}}$ are reported in the following sections. Note that only the results are shown since the demonstration requires long and tedious algebraic manipulations.

Using these optimal coefficients,(2.83)-(2.84) for $\mathbf{lsb_q} = \mathbf{lsb_{IC}}$ and (2.85)-(2.86) for $\mathbf{lsb_q} = \frac{1}{2}\mathbf{lsb_{IC}}$, one obtains two different *Linear Minimum means Square error* functions, called LMS1b and LMS2b respectively. In Fig. 2.12 there is the comparison between the value of $f_{opt}(\text{IC})$, $f_{lin}(\text{IC})$,$f_{\text{LMS1b}}(\text{IC})$ and $f_{\text{LMS2b}}(\text{IC})$ for all the possible input values ($n = 6$, $h = 0$), collected according to the IC value. Note that for some IC values the $f_{\text{LMS2b}}(\text{IC})$ is closer to $f_{lin}(\text{IC})$ than $f_{\text{LMS1b}}(\text{IC})$ (zoom A in Fig. 2.12), for others $f_{\text{LMS1b}}(\text{IC})$ is closer to $f_{lin}(\text{IC})$ than $f_{\text{LMS2b}}(\text{IC})$ (zoom B in Fig. 2.12), but in average, as it will be demonstrated in the following $f_{\text{LMS2b}}(\text{IC})$ gives the best approximation of the $f_{lin}(\text{IC})$ and hence of $f_{opt}(\text{IC})$.

Quantization with 1bit $\mathbf{lsb_q} = \mathbf{lsb_{IC}}$ (LMS1b function)

The optimal *qi* values which solve the problem (2.81) are:

$$\begin{array}{l} q_1 = q_2 = q_{n_{eq}-1} = q_{n_{eq}} = 1 \\ q_3 = q_4 = ... = q_{n_{eq}-2} = 2 \end{array} \quad (\text{for } \mathbf{lsb_q} = \mathbf{lsb_{IC}}) \tag{2.83}$$

By substituting these coefficients in (2.82) we found:

$$K_{\text{round}} = \frac{\mathbf{lsb}}{2} \quad (\text{for } \mathbf{lsb_q} = \mathbf{lsb_{IC}}) \tag{2.84}$$

From (2.83), it can be observed that the quantized coefficients q_i are equal to the coefficients l_i of the linear function $f_{lin}(\text{IC})$ rounded to the nearest integer[1] (see also Fig. 2.11).

Quantization with 2bit $\mathbf{lsb_q} = \frac{1}{2}\mathbf{lsb_{IC}}$ (LMS2b function)

The values of K_{round} and $q_2, \ldots q_{n_{eq}-1}$ which solve the problem (2.81) are:

$$\begin{array}{l} q_2 = q_3 = ... = q_{n_{eq}-1} = 1.5 \\ Kr = \dfrac{\mathbf{lsb}}{2} + \dfrac{\mathbf{lsb_{IC}}}{2}\left(\left\lfloor \dfrac{n_{eq}}{4} \right\rfloor - 1\right) \end{array} \quad (\text{for } \mathbf{lsb_q} = 1/2\mathbf{lsb_{IC}}) \tag{2.85}$$

[1] Please note that, from (2.63), $l_2 = l_{n_{eq}-1} = \frac{3}{2} - \frac{4}{3}2^{-n_{eq}} < 1.5$

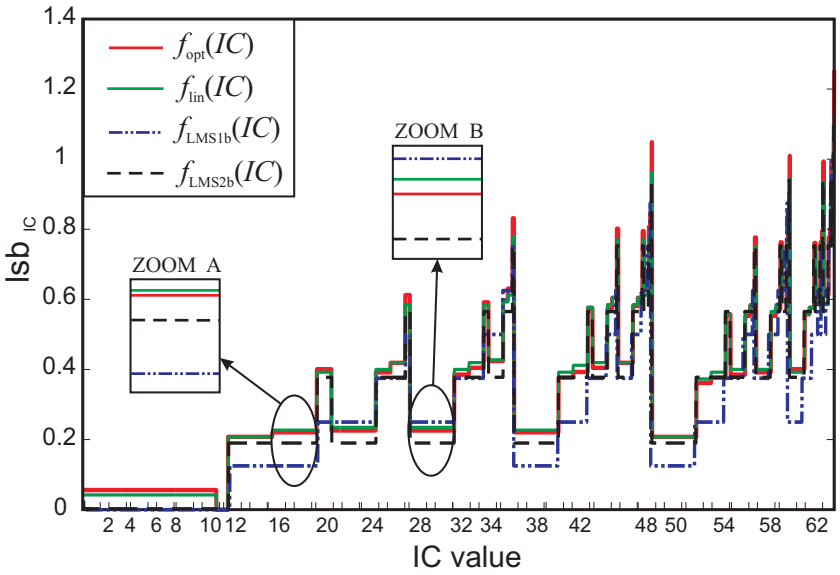

Figure 2.12: Comparison between the optimal error compensation function f_{opt}(IC), red full line, f_{lin}(IC), green full line, f_{LMS1b}(IC), blue dashed-dotted line, f_{LMS2b}(IC), black dashed line.

for every value of *n*and *h*that verify $n_{eq} > 3$. Multiple solutions exist for q_1 and $q_{n_{eq}}$. The solution that we will consider in the following (for $n_{eq} > 3$) is:

$$\begin{array}{llll} q_1 = 1.0; & q_{n_{eq}} = 1.0 & \text{if REM}(n_{eq}, 4) & = 0 \\ q_1 = 1.0; & q_{n_{eq}} = 1.5 & \text{if REM}(n_{eq}, 4) & = 1 \\ q_1 = 1.5; & q_{n_{eq}} = 1.5 & \text{if REM}(n_{eq}, 4) & = 2 \\ q_1 = 1.5; & q_{n_{eq}} = 1.5 & \text{if REM}(n_{eq}, 4) & = 3 \end{array} \quad (\mathbf{lsb_q} = \tfrac{1}{2}\mathbf{lsb_{IC}}) \tag{2.86}$$

where REM(*a*,*b*) indicates the remainder of the integer division *a*/*b*. From (2.86) we note that, in this case, the quantized coefficients q_i corresponds to the coefficients l_i rounded to the nearest integer only when REM(*neq*,4) is equal to 2 or 3.

2.6 Mean Square Error

The expression of the total mean square error for truncated multiplier is:

$$\varepsilon^2_{\text{total}} \simeq \sigma^2_{\text{trunc}} + \varepsilon^2_{\text{intrinsic}} + \sigma^2_{\text{lin}} + \sigma^2_{\text{q}} + \left(\mu_{\text{quant}} + \mu_{\text{trunc}} + K_{\text{round}}\right)^2 \tag{2.87}$$

where:

σ^2_{trunc} (2.26) is the variance of the truncation error, which arises from using only n output bits of the result;

$\varepsilon^2_{\text{intrinsic}}$ (2.57) is the intrinsic error, independent on the compensation function, which arises from using only the vector IC in order to estimate the matrix $\text{LSP}_{\text{minor}}$;

σ^2_{lin} (2.68) is the variance of the compensation error, which arises from using a linear approximation of $f_{opt}(\text{IC})$;

σ^2_{q} and μ_{quant} are the variance and the mean of $\Delta f_{lin}(\text{IC})$, which arises from the quantization of the coefficient;

K_{round} is the rounding constant.

All the components are note with the exception of σ^2_{q} and μ_{quant}, which can be computed using(2.71):

$$\Delta l_i = q_i - l_i \tag{2.88}$$

where l_i are the coefficient of the linear function (2.63) while q_i is the coefficient of the LMS compensation function, which depend on the chosen quantization. Let's see the exact value of these errors, and hence of the $\varepsilon^2_{\text{total}}$ considering the two different quantizations.

2.6.1 Analytical calculation of $\varepsilon^2_{\text{total}}$ for LMS1b function

Using the expression of the coefficients q_i (2.83) in (2.78) and (2.75), the two terms σ^2_{q} and μ_{total}, can be expressed as:

$$\sigma^2_{\text{q}} = \mathbf{lsb}^2 \cdot 2^{-2h-2} \cdot \left[\frac{7}{72} - \frac{5}{3}2^{-n_{eq}} - \frac{1}{18}2^{-2n_{eq}} + \frac{1}{12}\left(2^{-n_{eq}} + \frac{1}{4}\right) n_{eq}\right] \tag{2.89}$$

$$\mu_{\text{total}} = \mathbf{lsb_{IC}} \cdot \left(-2^{-n_{eq}-1}\right) \tag{2.90}$$

Hence all the component of the error are known and the final expression of the error can be easily computed.

Note that (2.87) is approximated since it does not consider correlation between e_{erasing} and e_{trunc} (see Sec. 2.2). It is possible to improve the accuracy by considering part of the correlation existing between μ_{erasing} and e_{trunc}. This

result will be reported only for the case $\mathbf{lsb_q} = \mathbf{lsb_{IC}}$, but it will be shown that the hypothesis of independence is quite good, since the simulation results gives the same results of the analytical formula.

The major contribution of the correlation between e_{erasing} and e_{trunc} exists on the column with the weight $\mathbf{lsb_q} = \mathbf{lsb_{IC}}$. In fact this column is the only one in which we have only terms belonging to $f(\text{IC})$, (see Fig. 2.1). These terms are, of course, strongly correlated to the terms of the IC. In order to take into account this correlation we can split the truncation error e_{trunc} in two contributions:

$$e_{\text{trunc}} = e_{\text{truncH}} + e_{\text{truncL}} \tag{2.91}$$

where (consider that, for $\mathbf{lsb_q} = \mathbf{lsb_{IC}}$, *m*=*h*+1):

$$e_{\text{truncH}} = -\sum_{i=n+1}^{n+h} pt_i \cdot 2^{-i} \quad ; \quad e_{\text{truncL}} = -pt_{n+h+1} \cdot 2^{-n-h-1} \tag{2.92}$$

In the following the correlation between e_{erasing} and e_{truncL} will be considered and the more complicated correlation between e_{erasing} and e_{truncH} will be neglected.

With position (2.91) the total mean square error can be written as:

$$\begin{aligned} \varepsilon^2_{\text{total}} &= E\left\{\left(K_{\text{round}} + e_{\text{erasing}} + e_{\text{truncL}} + e_{\text{truncH}}\right)^2\right\} \simeq \\ &\simeq E\left\{\left(K_{\text{round}} + e_{\text{erasing}} + e_{\text{truncL}} + \mu_{\text{truncH}}\right)^2\right\} + \sigma^2_{\text{truncH}} \end{aligned} \tag{2.93}$$

where the hypothesis of independence between e_{erasing} and e_{truncH} have been exploited. The variance σ^2_{truncH} can be computed by assuming the bits of e_{truncH} independent and equally likely:

$$\sigma^2_{\text{truncH}} = \sum_{i=n+1}^{n+h} 2^{-2i-2} = \mathbf{lsb}^2 \frac{1}{12}\left(1 - 2^{-2h}\right) \tag{2.94}$$

The first term in (2.93) can be easily evaluated taking into account the correlation between e_{erasing} and e_{truncL}:

$$\begin{aligned} &E\left\{\left(K_{\text{round}} + e_{\text{erasing}} + e_{\text{truncL}} + \mu_{\text{truncH}}\right)^2\right\} = \\ =&E\left\{\left(K_{\text{round}} + e_{\text{erasing}} + \mu_{\text{truncH}}\right)^2\right\} + \varepsilon^2_{\text{truncL}} + \\ &+2 \cdot E\left\{\left(K_{\text{round}} + e_{\text{erasing}} + \mu_{\text{truncH}}\right) \cdot e_{\text{truncL}}\right\} \end{aligned} \tag{2.95}$$

The first term in (2.95), following the theory presented in previous sections, can be written as:

$$E\left\{\left(K_{\text{round}}+e_{\text{erasing}}+\mu_{\text{truncH}}\right)^2\right\}= \\ =\varepsilon^2_{\text{intrinsic}}+\sigma^2_{\text{lin}}+\sigma^2_{\text{q}}+\left(K_{\text{round}}+\mu_{\text{erasing}}+\mu_{\text{truncH}}\right)^2 \tag{2.96}$$

The last term in (2.95) is a correlation coefficient ρ_L:

$$\rho_L=2\cdot E\left\{\left(K_{\text{round}}+e_{\text{erasing}}+\mu_{\text{truncH}}\right)\cdot e_{\text{truncL}}\right\} \tag{2.97}$$

In conclusion we have:

$$\varepsilon^2_{\text{total}}\simeq\varepsilon^2_{\text{intrinsic}}+\sigma^2_{\text{lin}}+\sigma^2_{\text{q}}+\left(K_{\text{round}}+\mu_{\text{erasing}}+\mu_{\text{truncH}}\right)^2+ \\ +\varepsilon^2_{\text{truncL}}+\rho_L+\sigma^2_{\text{truncH}} \tag{2.98}$$

Some of these components have already been evaluated: $\varepsilon^2_{\text{intrinsic}}$, σ^2_{lin}, σ^2_{q} and σ^2_{truncH} are given by (2.57), (2.68), (2.89) and (2.94) respectively. The other components have been analytically evaluated. For brevity we report only the results:

$$\left(K_{\text{round}}+\mu_{\text{erasing}}+\mu_{\text{truncH}}\right)^2=\mathbf{lsb}^2\cdot 2^{-2h-2}\left(\frac{1}{4}-\frac{1}{2}2^{-n_{eq}}+\frac{1}{4}2^{-2n_{eq}}\right) \tag{2.99}$$

$$\varepsilon^2_{\text{truncL}}=\begin{cases}\mathbf{lsb}^2\cdot 2^{-2}\dfrac{17}{32} & h=0\\ \mathbf{lsb}^2\cdot 2^{-2h-2}\cdot\dfrac{15}{32} & h>0\end{cases} \tag{2.100}$$

$$\rho_L=\begin{cases}\mathbf{lsb}^2\cdot 2^{-2h-2}\left(-\dfrac{19}{32}+2^{-n_{eq}}\right) & h=0\\ \mathbf{lsb}^2\cdot 2^{-2h-2}\cdot\left(-\dfrac{13}{32}\right) & h>0\end{cases} \tag{2.101}$$

It is worth to highlight that the approximations done in this new formula are those used for (2.93), when the contribution of e_{truncH} is disjoined. For h=0, e_{truncH}=0, and the formula is exact.

Summing all error components, we have the following expression of $\varepsilon^2_{\text{total}}$:

$$\frac{\varepsilon^2_{\text{total}}}{\mathbf{lsb}^2}\simeq\begin{cases}\dfrac{1}{12}+\left(-\dfrac{19}{576}+\dfrac{1}{48}n_{eq}-\dfrac{1}{4}2^{-n_{eq}}+\dfrac{1}{36}2^{-2n_{eq}}\right) & h=0\\ \dfrac{1}{12}+2^{-2h}\cdot\left(-\dfrac{1}{576}+\dfrac{1}{48}n_{eq}-\dfrac{1}{2}2^{-n_{eq}}+\dfrac{1}{36}2^{-2n_{eq}}\right) & h>0\end{cases} \tag{2.102}$$

Note that, as highlighted before, this expression gives exactly the mean square error $\varepsilon^2_{\text{total}}$ for $h = 0$. As we will show in Sec. 2.6.3, the expression (2.102) is a very good approximation for $h > 0$.

The expression (2.102) can be approximated by neglecting negative exponential terms:

$$\frac{\varepsilon^2_{\text{total}}}{\mathbf{lsb}^2} \simeq \begin{cases} \dfrac{1}{12} + \left(-\dfrac{19}{576} + \dfrac{1}{48} n_{eq}\right) & h = 0 \\ \dfrac{1}{12} + 2^{-2h} \cdot \left(-\dfrac{1}{576} + \dfrac{1}{48} n_{eq}\right) & h > 0 \end{cases} \tag{2.103}$$

Employing (2.103) in place of (2.102) introduces only a small increase in the mean square error, lower than 5.4% for $n_{eq} \geq 5$.

2.6.2 Analytical calculation of $\varepsilon^2_{\text{total}}$ for LMS2b function

As done for LMS1b implementation, the resulting σ^2_{q} and μ_{total} are:

$$\begin{aligned} \sigma^2_{\text{q}} = \mathbf{lsb}^2 \cdot 2^{-2h-2} \cdot \Bigg[& \frac{1}{288} - \frac{1}{6} 2^{-n_{eq}} - \frac{1}{18} 2^{-2n_{eq}} \\ & + \frac{1}{12} \left(2^{-n_{eq}} + \frac{1}{16}\right) n_{eq} + \left(-\frac{1}{64} + \frac{1}{8} 2^{-n_{eq}}\right) \cdot \zeta\left(n_{eq}\right) \Bigg] \end{aligned} \tag{2.104}$$

$$\mu_{\text{total}} = \mathbf{lsb_{IC}} \cdot \left(-\frac{1}{2} 2^{-n_{eq}} - \frac{\text{REM}(n_{eq}, 4) - \zeta\left(n_{eq}\right)}{8}\right) \tag{2.105}$$

where the function $\zeta(n)$ is defined as:

$$\zeta(n) = \begin{cases} \text{REM}(n, 4) & \text{if : } \text{REM}(n, 4) = 0, 1, 2 \\ 2 & \text{if : } \text{REM}(n, 4) = 3 \end{cases} \tag{2.106}$$

By adding all error components we can compute the following expression of the total mean square error of the multiplier with $\mathbf{lsb_q} = 1/2\mathbf{lsb_{IC}}$:

$$\frac{\varepsilon^2_{\text{total}}}{\mathbf{lsb}^2} \simeq \begin{cases} \frac{1}{12} + 2^{-2h} \cdot \left(-\frac{58 + 9\zeta\left(n_{eq}\right)}{2304} + \frac{13}{768} n_{eq} + \right. & \text{REM}\left(n_{eq}, 4\right) \leq 2 \\ \left. + \frac{\zeta\left(n_{eq}\right)}{32} 2^{-n_{eq}} + \frac{1}{36} 2^{-2n_{eq}} \right) & \\ \frac{1}{12} + 2^{-2h} \cdot \left(-\frac{67}{2304} + \frac{13}{768} n_{eq} + \right. & \text{REM}\left(n_{eq}, 4\right) = 3 \\ \left. + \frac{3}{32} 2^{-n_{eq}} + \frac{1}{36} 2^{-2n_{eq}} \right) & \end{cases} \tag{2.107}$$

The expression (2.107) can be approximated by neglecting negative exponential terms and assuming $\zeta(n_{eq}) \simeq 1$:

$$\frac{\varepsilon^2_{\text{total}}}{\mathbf{lsb}^2} \simeq \frac{1}{12} + 2^{-2h} \cdot \left(-\frac{67}{2304} + \frac{13}{768} n_{eq} \right) \tag{2.108}$$

Employing (2.108) in place of (2.107) introduces only a small variation in the calculated mean square error, lower than 2.8% for $n_{eq} \geq 5$.

2.6.3 Results

Having closed form expressions, it is possible to compute quickly the mean square error for every value of *n* and *h*. This makes very simple to design the multiplier for a given *n* by choosing *h* on the basis of the required accuracy. This approach can hardly be employed in previously proposed variable correction truncated multipliers, where simulations are needed to compute the error.

If we compare the approximated expression of $\varepsilon^2_{\text{intrinsic}}$ (see (2.58)), with the approximated expressions of $\varepsilon^2_{\text{total}}$ for $\mathbf{lsb_q} = \mathbf{lsb_{IC}}$ (see (2.103)) and $\mathbf{lsb_q} = 1/2\mathbf{lsb_{IC}}$ (see (2.108)), we can observe that the intrinsic error $\varepsilon^2_{\text{intrinsic}}$ (and also the error $\varepsilon^2_{\text{low\_bound}} \simeq \sigma^2_{\text{trunc}} + \varepsilon^2_{\text{intrinsic}}$) increases with $0.015 \cdot n_{eq} \cdot \mathbf{lsb}^2 \cdot 2^{-2h}$ while the error (2.103) of the proposed multiplier with $\mathbf{lsb_q} = \mathbf{lsb_{IC}}$ increases with $0.021 \cdot n_{eq} \cdot \mathbf{lsb}^2 \cdot 2^{-2h}$. From (2.108) the error $\varepsilon^2_{\text{total}}$ of the proposed multiplier with $\mathbf{lsb_q} = 1/2\mathbf{lsb_{IC}}$ increases with $0.017 \cdot n_{eq} \cdot \mathbf{lsb}^2 \cdot 2^{-2h}$. The error of the proposed multiplier with $\mathbf{lsb_q} = 1/2\mathbf{lsb_{IC}}$, therefore, is very close the lower bound $\varepsilon^2_{\text{low\_bound}}$.

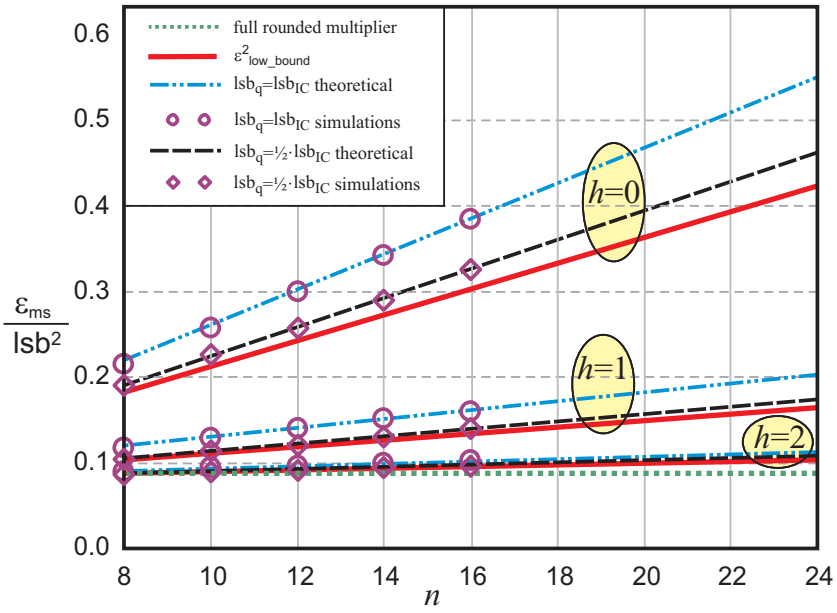

Figure 2.13: Comparison between total mean square errors. Theoretical values are computed by using the formulae described in the chapter.

Fig. 2.13 shows the total mean square error ($\varepsilon^2_{\text{total}}$) obtained for $\mathbf{lsb_q} = \mathbf{lsb_{IC}}$ and $\mathbf{lsb_q} = 1/2\mathbf{lsb_{IC}}$. The figure plots both theoretical values and simulation results. The simulation time needed to obtain the mean square error increases as O(2^{2n}), therefore simulation results are only presented for $n \leq 16$.

For comparison the figure also shows the minimum mean square error $\varepsilon^2_{\text{low\_bound}}$ achievable with a variable correction truncated multiplier (2.44) and the error of a full-rounded multiplier ($\sigma^2_{\text{trunc}} = \mathbf{lsb}^2/12$). As it can be seen, the theoretical values for $\mathbf{lsb_q} = \mathbf{lsb_{IC}}$ are almost exact for any h value, the maximum difference between theoretical and simulation values being lower than $10^{-3}\mathbf{lsb}^2$. For $\mathbf{lsb_q} = 1/2\mathbf{lsb_{IC}}$ the theoretical values are very close to simulations for $h \geq 1$. For $h = 0$ the maximum difference between theoretical and simulated values is $7 \cdot 10^{-3}\mathbf{lsb}^2$. From Fig. 2.13, we note again that the mean square error obtained with $\mathbf{lsb_q} = 1/2\mathbf{lsb_{IC}}$ is close to the error $\varepsilon^2_{\text{low\_bound}}$. Thus, from a practical point of view, the proposed truncated multiplier obtains the best performances achievable with a variable error correction approach.

Data in Fig. 2.13 demonstrate that the hypothesis of independence between truncation and compensation errors, although not exactly verified, represent a reasonable assumption.

2.7 Maximum Absolute Error

Using (2.16) the maximum absolute error $\varepsilon_{\max}$ of the LMS1b truncated multiplier is:

$$\varepsilon_{\max} = \max_{x,y} \left(\left| f(\mathrm{IC}) - S_{\mathrm{LSP_{minor}}} + e_{\mathrm{trunc}} + K_{\mathrm{round}} \right| \right) \tag{2.109}$$

where e_{trunc} is the truncation error given by (2.24). Note that the maximum absolute error cannot be easily computed for a generic truncated multiplier due to the dependency of the truncation error on the compensation function. In the particular case of the LMS1b circuit proposed, $\varepsilon_{\max}$ can be analytically computed for every n and h value.

Defining AP, the matrix obtained considering $\mathrm{LSP_{minor}}$ without IC (Fig. 2.1), $S_{\mathrm{LSP_{minor}}}$ (2.10), can be divided into two parts:

$$S_{\mathrm{IC}} = 2^{-n-h-1} \sum_{i=1}^{n_{eq}} \gamma_i \tag{2.110}$$

$$S_{\mathrm{AP}} = \sum_{i=h+2}^{n} \sum_{j=n+h+2-i}^{n} x_i y_j 2^{-i-j} \tag{2.111}$$

where S_{IC} is the weighted sum of the IC elements, S_{AP} the weighted sum of AP. Hence the maximum absolute error is obtained choosing the maximum absolute value between:

$$\varepsilon^{-} = \min_{x,y} \left(\left(f(\mathrm{IC}) - S_{\mathrm{IC}} \right) - S_{\mathrm{AP}} + e_{\mathrm{trunc}} + K_{\mathrm{round}} \right) \tag{2.112}$$

$$\varepsilon^{+} = \max_{x,y} \left(\left(f(\mathrm{IC}) - S_{\mathrm{IC}} \right) - S_{\mathrm{AP}} + e_{\mathrm{trunc}} + K_{\mathrm{round}} \right) \tag{2.113}$$

Recalling that $e_{\mathrm{trunc}} \in \{-\mathbf{lsb} + \mathbf{lsb_q}, 0\}$, when ε^{-} is considered e_{trunc} must be chosen equal to the lowest value, $-\mathbf{lsb} + \mathbf{lsb_q}$, since we want to minimize (2.112). When we are computing ε^{+}, since we want to maximize (2.113), e_{trunc} must be chosen equal to the highest value, 0.

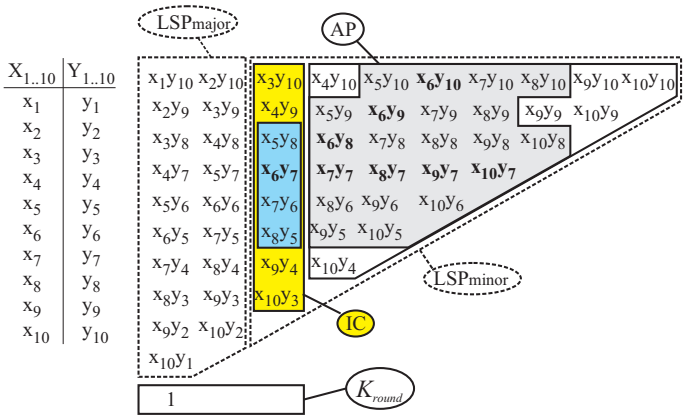

Figure 2.14: Correlations between the terms of the IC and the AP ($\text{LSP}_{\text{minor}} - \text{IC}$. The gray part of the AP is correlated with the blue part of the IC. As example the partial products shown in bold depend on x_6 and y_7. If $x_6 y_6 = 0$ this means that the bold row or the bold diagonal or both are identically equal to zero.

Using the definition of LMS1b function proposed in Sec. 2.5, the compensation function is:

$$f_{\text{LMS1b}}(\text{IC}) = \left[\gamma_1 + \gamma_2 + \gamma_{n_{eq}-1} + \gamma_{n_{eq}} + 2 \cdot \sum_{i=3}^{n_{eq}-2} \gamma_i \right] \cdot \mathbf{lsb_{IC}} \tag{2.114}$$

hence:

$$(f(\text{IC}) - S_{\text{IC}}) = \mathbf{lsb_{IC}} \cdot \sum_{i=3}^{n_{eq}-2} \gamma_i \tag{2.115}$$

The rounding constant is given by:

$$K_{\text{round}} = \frac{\mathbf{lsb}}{2} \tag{2.116}$$

hence:

$$\varepsilon^- = \min_{x,y} \left(\mathbf{lsb_{IC}} \cdot \sum_{i=3}^{n_{eq}-2} \gamma_i - S_{\text{AP}} - \frac{\mathbf{lsb}}{2} + \mathbf{lsb_{IC}} \right) \tag{2.117}$$

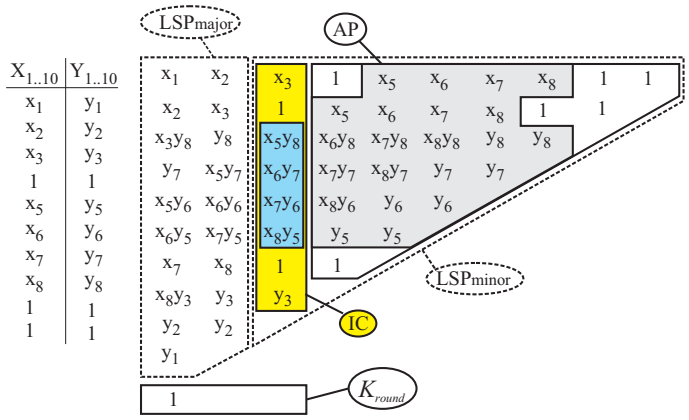

Figure 2.15: $\text{LSP}_{\text{minor}}$ partial product matrix for 10×10 bit multiplier with $h = 2$. The partial products of the AP that are uncorrelated with the central terms of the IC have been fixed to 1.

$$\varepsilon^{+} = \max_{x,y} \left(\mathbf{lsb_{IC}} \cdot \sum_{i=3}^{n_{eq}-2} \gamma_i - S_{\text{AP}} + \frac{\mathbf{lsb}}{2} \right) \tag{2.118}$$

In oder to compute the maximum absolute error, (2.117) and (2.118) should be expressed in closed form. These computations, however, are not trivial.

Fig. 2.14 shows the least significant part of the partial product matrix and the IC of a 10×10 bit multiplier, with $h = 2$. The grey partial products of the AP have at least one bit shared with the central part of the IC that is involved in (2.117) and (2.118). As a consequence the value of S_{AP} and the value of $\mathbf{lsb_{IC}} \cdot \sum_{i=3}^{n_{eq}-2} \gamma_i$ cannot be independently chosen to calculate (2.117) and (2.118). On the contrary, since the extreme terms of the IC, γ_i ($i = \{1, 2, n_{eq} - 1, n_{eq}\}$), are not involved in (2.117) and (2.118), they can be chosen in order to independently fix e_{trunc}.

Minimum value of the punctual error ε^{-}

The minimum value ε^{-} (2.117) is obtained maximizing the AP partial products that are equal to one while maximizing the central partial products of the IC that are equal to zero. In the following we will refer to the case in which n_{eq} is

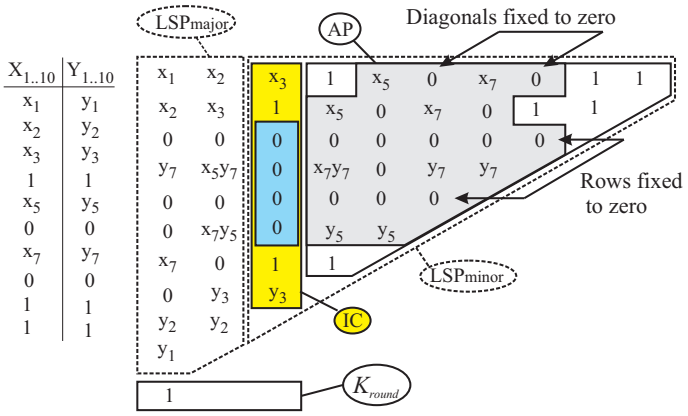

Figure 2.16: $\text{LSP}_{\text{minor}}$ partial product matrix for 10×10 bit multiplier with $h = 2$. The central terms of the IC are fixed to zero. Each γ_i term fixed to zero fixes one row and one diagonal of the AP to zero, alternatively.

even. The extension to the odd case will be provided at the end of the Section.

Figures (2.15), (2.16), and (2.17) show the step by step modification of the AP and of the IC during the following algorithm that maximizes the punctual error. They will also show how the input bits that determine the minimum error condition are calculated.

Firstly, the value of every term of the AP that is not in the grey zone can be chosen equal to 1 in order to maximize S_{AP}. This means fixing to 1 the input bits x_{h+2},x_{n-1},x_n,y_{h+2},y_{n-1}, and y_n. As a consequence the IC terms $x_{h+2}y_{n-1}$ and $y_{h+2}x_{n-1}$ are also 1. The partial product matrix for $n = 10$ and $h = 2$ is modified as shown in Fig. 2.15.

The next step in minimizing the punctual error is fixing to zero the central terms of the IC that is: $\gamma_i = 0$, $i = \{3, \ldots, n_{eq} - 2\}$. When the single term $\gamma_i = x_{h+i}y_{n+1-i}$ is fixed to zero, a whole row of AP (if $y_{n+1-i} = 0$) or a whole diagonal of AP (if $x_{h+i} = 0$) or both (if $y_{n+1-i} = x_i = 0$) are equal to zero. Since the target is having as many terms of the AP equal to one as possible, it is assumed that each zero γ_i fixes to zero only one row or one diagonal of the AP.

It is worth demonstrating that posing to zero the central γ_i minimizes the punctual error. This can be done noting that posing $\gamma_i = 1$ increases (2.117)

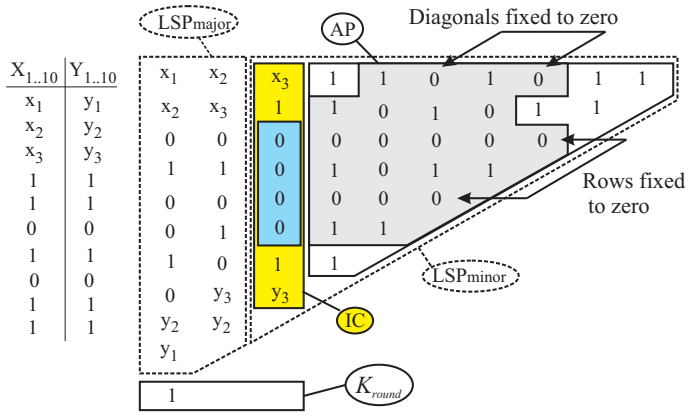

Figure 2.17: $\text{LSP}_{\text{minor}}$ partial product matrix for 10×10 bit multiplier with $h = 2$. The remaining available bits of the AP shown in Fig. 2.16 are fixed to 1.

by $\mathbf{lsb_{IC}}/2$. In order to compensate this increment the complete row or the complete diagonal of the AP can be posed equal to 1. However, one whole row or diagonal of AP equal to 1 decreases (2.117) by $\frac{\mathbf{lsb_{IC}}}{2} \sum_{j=1}^{n_{eq}-1} \frac{1}{2^j}$ that is lower than $\frac{\mathbf{lsb_{IC}}}{2}$.

For each γ_i it is therefore necessary to decide if it is more convenient to fix to zero the correlated row or the correlated diagonal.

To that purpose truncated multipliers with different bitwidths have been designed and the analysis has shown that, despite the bitwidth of the multiplier, (2.117) is always minimized according to the following conjecture:

Conjecture

when n_{eq} is even (2.117) is minimized when

x_{h+4}, x_6,..., and x_{n-2} are equal to 0

y_{h+4}, y_6,..., and y_{n-2} are equal to 0

when n_{eq} is odd (2.117) is minimized when

x_{h+4}, x_{h+6},..., x_{n-3} and x_{n-2} are equal to 0

y_{h+5}, y_{h+7},..., and y_{n-2} are equal to 0

It's worth highlighting that notwithstanding the conjecture is easily ver-

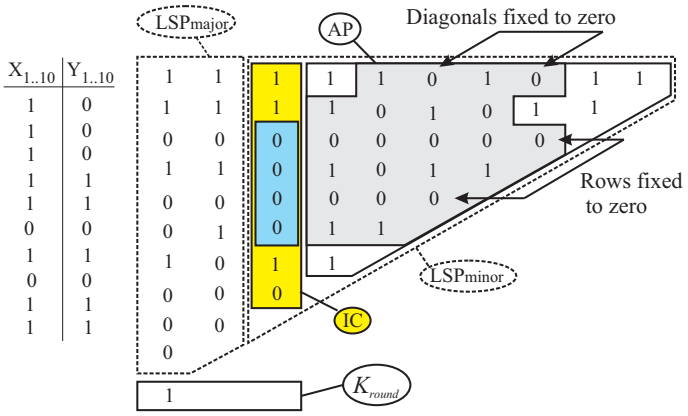

Figure 2.18: One of the possible configurations of the $\text{LSP}_{\text{minor}}$ for a 10×10 bit LMS1b truncated multiplier with $h = 2$ that maximizes the punctual error.

ified for a particular n_{eq} value, but it is not demonstrated for a general n_{eq} bit multiplier. The truth of the conjecture is verified in the following through numerical simulations. For example it is valid for $n = 10$, $h = 0$ and the resulting matrix is shown in Fig. 2.16, where for $\gamma_3 = 0$ the correlated row has been fixed to zero ($y_8 = 0$), for $\gamma_4 = 0$ the correlated diagonal has been fixed to zero ($x_6 = 0$), and so on.

The remaining terms of the AP matrix are now independent on the IC and hence have to be fixed to 1 in order to maximize the punctual error. The resulting matrix is shown in Fig. 2.17.

Finally the $x_1, \ldots, x_{h+1}$ and $y_1, \ldots, y_{h+1}$ bits are chosen minimizing e_{trunc}, hence imposing an odd number of 1 in each column of $\text{LSP}_{\text{major}}$ and IC. Remember that in $(n+h)^{th}$ column there is already a bit equal to 1 given by K_{round}.

x_{h+1} and y_{h+1} must be chosen in order to have an odd number of bits equal to 1 in IC column. Until now the IC vector has 2 bits equal to 1. Hence when $h = 0$, due to the presence of K_{round}, $x_{h+1} = y_{h+1}$; when $h \neq 0$, the choice must be $x_{h+1} \neq y_{h+1}$ (see Fig. 2.17). Finally also $x_1, \ldots, x_h$ and $y_1, \ldots, y_h$ must be also chosen in order to have all 1 in the bits of the sum (so the truncation error is minimum). The configuration is not simple neither unique since one must consider also the carry of the sum of each column.

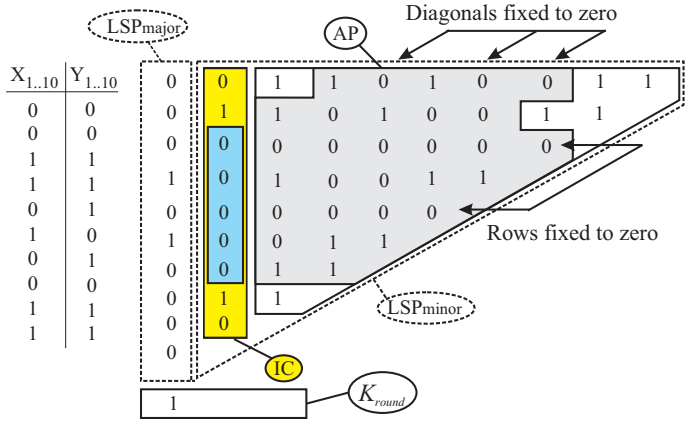

Figure 2.19: One of the possible configurations of the $\text{LSP}_{\text{minor}}$ for a 10×10 bit LMS1b truncated multiplier with $h = 1$ that maximizes the punctual error.

Note that the values of these bits are not required for the calculation of the expression in closed form of the maximum absolute error (see (2.109)). Figure 2.18 shows the values of the partial products of the $\text{LSP}_{\text{minor}}$ that minimize the punctual error, for a 10×10 bit LMS1b truncated multiplier with $h = 2$, with one of the possible configuration of the inputs.

The next step is calculating the punctual error for this particular configuration of the input bits. The IC component to the minimum punctual error is zero. Let's calculate the component provided by the sum of AP terms equal to 1.

The column whose weight is $\mathbf{lsb_{IC}}/4$, composed by $n_{eq} - 1$ elements has the first and the last terms equal to 1 while the other terms are an alternating sequence of 1 and 0. The contribution of this column to the punctual error is:

$$\frac{\mathbf{lsb_{IC}}}{4}\left(2 + \left\lceil \frac{n_{eq} - 3}{2} \right\rceil\right) = \frac{\mathbf{lsb_{IC}}}{8}(n_{eq} + 2) \tag{2.119}$$

The subsequent column, composed by $n_{eq} - 2$ terms, whose weight is $\mathbf{lsb_{IC}}/8$, has only the first and the last terms equal to 1. The contribution of this column to the punctual error is $\mathbf{lsb_{IC}}/4$.

The next column, composed by $n_{eq} - 3$ terms whose weight is $\mathbf{lsb_{IC}}/16$, is again alternatively composed by 0 and 1 elements. The contribution of this

column to the punctual error is $\frac{3}{16}\mathbf{lsb_{IC}}$. And so on.

In general, let's indicate with C_{ODD} the contribute of the odd column of $\text{LSP}_{\text{minor}}$ $\{IC, (n+h+3)^{th}$ column, $(n+h+5)^{th}, \ldots, (n+h+(2j-1))^{th}$; $j = 1, \ldots, n_{eq}/2\}$ and with C_{EVEN} the contribute of the even column of $\text{LSP}_{\text{minor}}$ $\{(n+h+2)^{th}$ column, $(n+h+4)^{th}, \ldots, (n+h+(2j))^{th}$; $j = 1, \ldots, n_{eq}/2\}$.

Since in odd columns there are present only two bits equal to 1:

$$C_{\text{ODD}} = \mathbf{lsb_{IC}} \left[\sum_{j=2}^{n_{eq}/2} 2 \cdot 2^{-2j+2} \right] \tag{2.120}$$

The even columns, except the first and the last, are alternatively composed by 0 and 1 elements; the first column is composed by 1 in first and last position and 1 alternate to 0 in the others; the last column is formed by only one 1. Hence:

$$\begin{aligned} C_{\text{EVEN}} = \mathbf{lsb_{IC}} \Bigg[& \left(3 + \frac{n_{eq}-4}{2}\right) \cdot 2^{-2} + 2^{-n_{eq}+1} + \\ & + \sum_{j=2}^{n_{eq}/2-1} \frac{n_{eq}-2j}{2} 2^{-2j+1} \Bigg] \end{aligned} \tag{2.121}$$

Using (2.120) and (2.121), the final expression of ε^- is:

$$\begin{aligned} \varepsilon^-_{even} &= \min_{x,y} \left(\mathbf{lsb_{IC}} \cdot \sum_{i=3}^{n_{eq}-2} \gamma_i - S_{\text{AP}} - \frac{\mathbf{lsb}}{2} + \mathbf{lsb_{IC}} \right) \\ &= -C_{\text{EVEN}} - C_{\text{ODD}} - \frac{\mathbf{lsb}}{2} + \mathbf{lsb_{IC}} \\ &= \mathbf{lsb} \cdot \left[-\frac{1}{18} 2^{-h} \left(7 + 3n_{eq} + 2^{1-n_{eq}}\right) + 2^{-h-1} - \frac{1}{2} \right] \end{aligned} \tag{2.122}$$

If n_{eq} is odd the partial product matrix that minimize the punctual error is slightly modified, according to the conjecture. The configuration for the case n=10 and $h = 1$ is shown in Fig. 2.19. There are always a number of partial products uncorrelated with the central terms of the IC that are posed to 1. Further, the central terms of the IC are 0 and fix to 0 one row or one diagonal of the AP. The only difference is that for the odd case the most convenient

choice is the one that for the last two terms of the central part of the IC fixes to zero always the diagonal. The calculus of the minimum error is very similar. The result for the odd n_{eq} case is:

$$\varepsilon^{-}_{odd} = \mathbf{lsb} \cdot \left[-\frac{1}{18} 2^{-h} \left(7 + 3n_{eq} + 2^{1-n_{eq}}\right) + 2^{-h-1} - \frac{1}{2} + \right. \\ \left. + \frac{1}{3} 2^{-h} \left(\frac{1}{8} + \frac{4}{3} 2^{1-n_{eq}} \right) \right] \tag{2.123}$$

The two expression of the minimum value of the punctual error can be grouped in only one:

$$\varepsilon^{-} = \mathbf{lsb} \cdot \left[-\frac{1}{18} 2^{-h} \left(7 + 3n_{eq} + 2^{1-n_{eq}}\right) + 2^{-h-1} - \frac{1}{2} \right. \\ \left. + \frac{1}{6} 2^{-h} \left(\frac{1}{8} + \frac{4}{3} 2^{1-n_{eq}} \right) - (-1)^{n_{eq}} \frac{1}{6} 2^{-h} \left(\frac{1}{8} + \frac{4}{3} 2^{1-n_{eq}} \right) \right] \tag{2.124}$$

Maximum value of the punctual error, ε^{+}

The calculation of the minimum value of the punctual error is similar to the previous section. The details are not reported here. The result is:

$$\varepsilon^{+} = \mathbf{lsb} \cdot \left[\frac{1}{18} \left(1 + 3n_{eq} + (-1)^{n_{eq}} \cdot 8 \cdot 2^{-n_{eq}}\right) \right] \tag{2.125}$$

Maximum absolute error

Since the absolute value of (2.125) is always lower than (2.124) we can state that the maximum absolute error, $\varepsilon_{\max}$, for the LMS1b truncated multiplier is:

$$\varepsilon_{\max} = \mathbf{lsb} \cdot \left[-\frac{1}{18} 2^{-h} \left(7 + 3n_{eq} + 2^{1-n_{eq}}\right) + 2^{-h-1} - \frac{1}{2} \right. \\ \left. + \frac{1}{6} 2^{-h} \left(\frac{1}{8} + \frac{4}{3} 2^{1-n_{eq}} \right) - (-1)^{n_{eq}} \frac{1}{6} 2^{-h} \left(\frac{1}{8} + \frac{4}{3} 2^{1-n_{eq}} \right) \right] \tag{2.126}$$

Note that the maximum absolute error is not provided by a trivial configuration of the inputs, like a configuration with $x = y = [1, 1, \cdots, 1]$ or $x = y = [0, 0, \cdots, 0]$. As example if we consider $n = 10$ the worst case x, y configuration is given by $x = y = [0, 1, 1, 0, 1, 0, 1, 0, 1, 1]$, if we consider $n = 9$ it is given by $x = [0, 1, 1, 0, 1, 0, 0, 1, 1]$ $y = [0, 1, 1, 1, 0, 1, 0, 1, 1]$. In fact the worst case error is provided by a x, y configuration that provides an

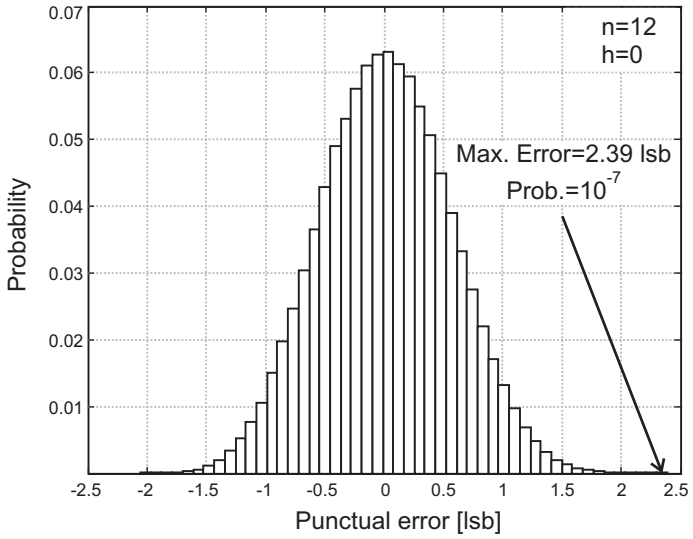

Figure 2.20: Probability distribution of the punctual error for a $n = 12$ bit LMS1b truncated multiplier, $h = 0$. The maximum absolute error is only present twice on the $2^{24} = 16.8 \times 10^6$ different inputs.

AP whose sum is badly approximated by the compensation function. Since the compensation function, on average, is a good approximation of the sum of the AP, such configuration is also the less probable to occur. Please note that the maximum absolute error of a truncated multiplier is only achieved for a very small number of input values. As an example in fig.2.20 the probability distribution of the punctual error for a 12 bit LMS1b truncated multiplier, $h = 0$, is shown. As can be seen the maximum absolute error has a very small probability (10^{-7}) whereas the error values having high probability are concentrated around zero. Hence non-exhaustive simulations are very unlikely to provide the actual maximum absolute error value or an error value near to the maximum. It is also worthwhile to note that, for each fixed h, the probability of the maximum absolute error decreases with n_{eq}, hence with n. In fact, the number of possible input configurations that give the maximum absolute error is two if n_{eq} is even or four if n_{eq} is odd (they are related to the fact that the same truncation error can be obtained choosing in two different ways the terms x_{h+1} and y_{h+1}). Hence, for increasing n_{eq}, the probability of the maximum absolute error decreases as $2^{-2n_{eq}+1}$ and $2^{-2n_{eq}+2}$ respectively.

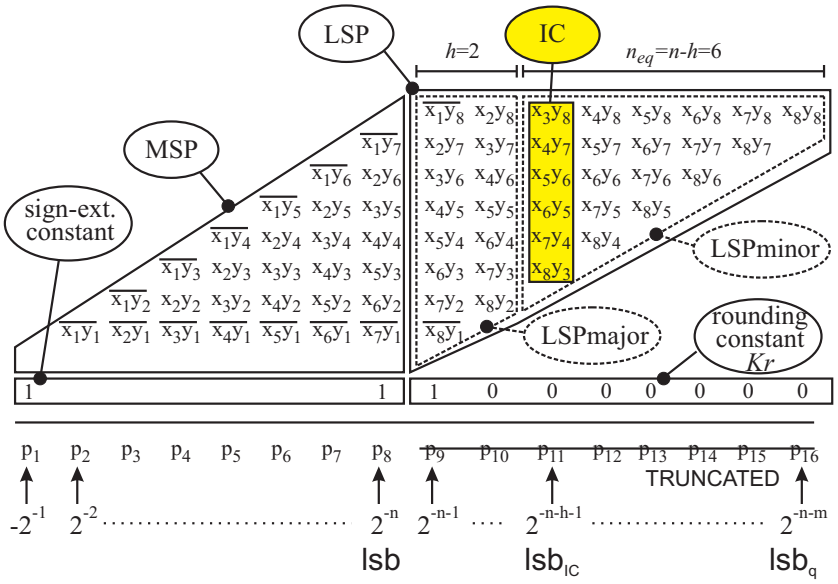

Figure 2.21: Partial Products matrix for a signed multipliers with $n = 8, h = 2$.

2.8 Signed and Mixed-Operand Multipliers

Fig. 2.21 shows the partial products matrix of a signed multiplier see Sec. 1.1.2. As it can be seen, when $h > 0$ the $\text{LSP}_{\text{minor}}$ part is equal to the $\text{LSP}_{\text{minor}}$ part of an unsigned multipliers. As a consequence, the expressions for the optimal compensation function (2.54), the intrinsic error (2.57), the linear compensation function (2.60), the quantized compensation function (2.70) and the total error statistics still hold. Therefore, in the following of this paragraph we will focus on the case $h = 0$, where two elements of the IC are complemented.

If we consider a signed-multiplier the IC vector is defined in the following way:

$$IC = [\gamma_1, \gamma_2, \ldots, \gamma_n] = [\overline{x_1 y_n}, x_2 y_{n-1}, \ldots, x_{n-1} y_2, \overline{x_n y_1}] \qquad (2.127)$$

Note that this definition of the IC assumes that the sign-extension prevention constant, shown in Fig. 2.21, is added to the partial products matrix.

Let us indicate as $fs_q(\text{IC})$ the linear compensation function with quantized

coefficients qs_i:

$$fs_q(\text{IC}) = 2^{-n-1}\left[\sum_{i=1}^{n} qs_i \cdot \gamma_i\right] \tag{2.128}$$

The following rule can be employed to transform the optimal coefficients q_i and rounding constant K_{round} given in previous sections for unsigned multiplier to the constant (Ks_{round}) and coefficients qs_i of the optimal signed multiplier (with $h = 0$):

$$\begin{aligned} qs_i &= 2 - q_i \qquad i = 1, n \\ qs_i &= q_i \qquad i \in \{2, ..., n-1\} \\ Ks_{round} &= K_{\text{round}} + \mathbf{lsb_{IC}} \cdot (q_1 + q_n - 2) \end{aligned} \tag{2.129}$$

Note that equation (2.129) is a valid transformation since it transforms quantized coefficients into quantized coefficients. By using (2.129), the total error statistics (μ_{total}, σ^2_{total} and $\varepsilon^2_{\text{total}}$) remains the same between unsigned and signed multipliers. The error formulas and analysis given in previous section, therefore, remains still valid also for signed multiplier.

The same reasoning can also be applied to mixed operand multipliers (signed×unsigned Sec. 1.1.3). Also in this case, for $h > 0$, the $\text{LSP}_{\text{minor}}$ correspond to the $\text{LSP}_{\text{minor}}$ of an unsigned multiplier. For $h > 0$ coefficients and error statistics are the same with respect to the unsigned multiplier.

For $h = 0$ the IC is defined as:

$$IC = [\gamma_1, \gamma_2, ..., \gamma_n] = [\overline{x_1 y_n}, x_2 y_{n-1}, ..., x_{n-1} y_2, x_n y_1] \tag{2.130}$$

For $h = 0$, the optimal quantized coefficients qm_i and rounding constant Km_{round} of the mixed operand multiplier can be obtained from the coefficients and constant q_i and rounding constant K_{round} of the unsigned multiplier by using the following transformation:

$$\begin{aligned} qm_1 &= 2 - q_1 \\ qm_i &= q_i \qquad i \in \{2, ..., n\} \\ Km_{round} &= K_{\text{round}} + \mathbf{lsb_{IC}} \cdot (q_1 - 1) \end{aligned} \tag{2.131}$$

Again, by using the transformation (2.131), the mixed operand multiplier achieves the same total error statistics with respect to the unsigned multiplier.

2.9 Conclusions

In this chapter a theoretical analysis of truncated multipliers with variable-correction has been presented.

A first result is showing that the optimal compensation function, which minimize the mean square error of a truncated multiplier, is a quadratic form of the partial products of the IC. It has been also evaluated, still in closed form, a lower bound for the error of any truncated multiplier designed using a variable correction methods.

The optimal compensation function, being a quadratic form, cannot be efficiently implemented in hardware. Therefore, it has been investigated the performance achievable by using a linear compensation function, best suited for hardware implementation. It is shown that the additional error component due to using a linear compensation function is negligible, pointing out that a linear compensation function is the best choice from a practical point of view.

The effect of coefficient quantization is also treated by providing the quantized optimal coefficients. Finally it has been computed, in closed form, the mean square error and the maximum absolute error of the LMS truncated multiplier. This is one of the most important characteristic of the proposed LSM multiplier. The LMS multiplier is the only architecture that can be designed, for every bit width, using an analytical approach that allows the a priori knowledge of the error committed. When no analytical approach is feasible, it can only be computed using slow exhaustive simulations, possible only for low n values.

The result given in the chapter are detailed for unsigned, signed and signed$\times$unsigned multipliers. All results can be applied for any n and h value. In the next chapter the implementation details of the proposed truncated multipliers will be discussed.

Chapter 3

VLSI implementation and Performances

The practical implementation of the quantized linear compensation function proposed in Ch. 2 is discussed in this chapter. The performances of the new truncated multipliers are extensively compared with previously proposed circuits. More than 100 truncated multipliers, with 8 different architectures have been synthesized in $0.18\mu m$ technology, with wordlengths ranging from 8 to 32 bits. Area, power and accuracy of the multipliers are investigated and compared. Experimental performances on a $0.18\mu m$ test chip are also presented. In following Sec. 3.1 describes the hardware implementation of truncated multipliers with quantized linear compensation function. The performances of the truncated multipliers are compared with previously proposed architectures in Sec. 3.2. Sec. 3.3 reports the experimental results obtained on the test chip implemented in $0.18\mu m$ CMOS technology.

3.1 Truncated Multipliers Implementations

As it will be shown in the following, the truncated multipliers based on a linear quantized compensation function proposed in Ch. 2, are efficiently implemented by summing a Partial Products Matrix (PPM). This is obtained by firstly using a carry-save reduction trees, followed by a fast carry-propagate adder (CPA) [22]. The Three Dimensional Minimization (TDM) method [5, 7, 6] will be exploited for the carry-save reduction tree; this tecnique minimizes the overall delay by compensating the delay asymmetries of full and half adders. In this section after a description of TDM technique proposed in

Table 3.1: Optimal quantized coefficient (Ch. 2). The $(\mathrm{REM}(x, y))$ symbol indicates the remainder of the integer division x/y.

Signed Multiplier	
LMS1b truncated multipliers ($\mathbf{lsb_q} = \mathbf{lsb_{IC}}$) any n , h values	$q_1 = q_2 = q_{n_{eq}-1} = q_{n_{eq}} = 1$ $q_3 = q_4 = \ldots = q_{n_{eq}-2} = 2$ $Kr = \mathbf{lsb}/2$
LMS2b truncated multipliers ($\mathbf{lsb_q} = \frac{1}{2}\mathbf{lsb_{IC}}$) $n_{eq} > 3$, $h = 0$	$q_1 = 1.0;\quad q_{n_{eq}} = 1.0 \quad$ if $\mathrm{REM}(n_{eq}, 4) = 0$ $q_1 = 1.0;\quad q_{n_{eq}} = 0.5 \quad$ if $\mathrm{REM}(n_{eq}, 4) = 1$ $q_1 = 0.5;\quad q_{n_{eq}} = 0.5 \quad$ if $\mathrm{REM}(n_{eq}, 4) = 2$ $q_1 = 0.5;\quad q_{n_{eq}} = 0.5 \quad$ if $\mathrm{REM}(n_{eq}, 4) = 3$ $q_2 = q_3 = \ldots = q_{n_{eq}-1} = 1.5$ $Kr = \frac{\mathbf{lsb}}{2} + \mathbf{lsb_q}\left(\lfloor \frac{n_{eq}}{4} \rfloor + 3 - 2q_1 - 2q_{n_{eq}}\right)$
LMS2b truncated multipliers ($\mathbf{lsb_q} = \frac{1}{2}\mathbf{lsb_{IC}}$) $n_{eq} > 3$, $h > 0$	$q_1 = 1.0;\quad q_{n_{eq}} = 1.0 \quad$ if $\mathrm{REM}(n_{eq}, 4) = 0$ $q_1 = 1.0;\quad q_{n_{eq}} = 1.5 \quad$ if $\mathrm{REM}(n_{eq}, 4) = 1$ $q_1 = 1.5;\quad q_{n_{eq}} = 1.5 \quad$ if $\mathrm{REM}(n_{eq}, 4) = 2$ $q_1 = 1.5;\quad q_{n_{eq}} = 1.5 \quad$ if $\mathrm{REM}(n_{eq}, 4) = 3$ $q_2 = q_3 = \ldots = q_{n_{eq}-1} = 1.5$ $Kr = \frac{\mathbf{lsb}}{2} + \mathbf{lsb_q}\left(\lfloor \frac{n_{eq}}{4} \rfloor - 1\right)$
Unsigned Multiplier	
LMS1b truncated multipliers ($\mathbf{lsb_q} = \mathbf{lsb_{IC}}$) any n , h values	$q_1 = q_2 = q_{n_{eq}-1} = q_{n_{eq}} = 1$ $q_3 = q_4 = \ldots = q_{n_{eq}-2} = 2$ $Kr = \mathbf{lsb}/2$
LMS2b truncated multipliers ($\mathbf{lsb_q} = \frac{1}{2}\mathbf{lsb_{IC}}$) $n_{eq} > 3$, $\forall h$	$q_1 = 1.0;\quad q_{n_{eq}} = 1.0 \quad$ if $\mathrm{REM}(n_{eq}, 4) = 0$ $q_1 = 1.0;\quad q_{n_{eq}} = 1.5 \quad$ if $\mathrm{REM}(n_{eq}, 4) = 1$ $q_1 = 1.5;\quad q_{n_{eq}} = 1.5 \quad$ if $\mathrm{REM}(n_{eq}, 4) = 2$ $q_1 = 1.5;\quad q_{n_{eq}} = 1.5 \quad$ if $\mathrm{REM}(n_{eq}, 4) = 3$ $q_2 = q_3 = \ldots = q_{n_{eq}-1} = 1.5$ $Kr = \frac{\mathbf{lsb}}{2} + \mathbf{lsb_q}\left(\lfloor \frac{n_{eq}}{4} \rfloor - 1\right)$

[5, 7, 6], the TDM approaches of [5] and [7] will be employed because they provide performances very close to the optimal TDM of [6] while providing a much simpler implementation. Recalling what has been told in Sec. 1.2.4 the

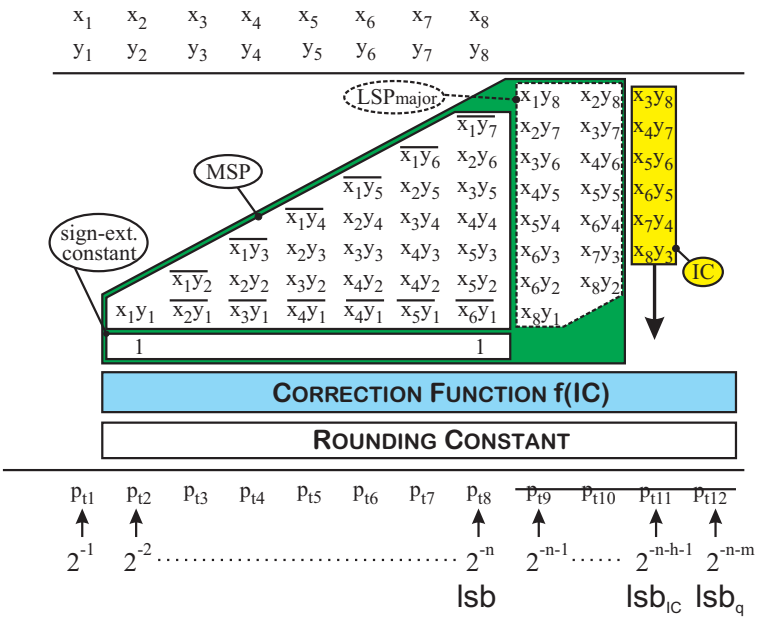

Figure 3.1: Signed truncated multiplier with $n = 8$, $h = 2$. In this example $\mathbf{lsb_q} = \frac{1}{2}\mathbf{lsb_{IC}}$

TDM of [7] is only composed by full adders providing the minimum number of terms to be summed in the final CPA adder. The approach of [7] provides fast multipliers, even if, in some rare cases, results in ripple structures that compromise the delay of the whole multiplier. The TDM of [5] uses both full and half adders, reducing the matrix to a height of two for every column. This avoids the problem of the carry ripple structures but requires a slightly more complex CPA.

Tab. 3.1 summarizes the optimal coefficient q_i and constant K_{round} computed in Ch. 2. As it can be seen, the quantized coefficients in the case $\mathbf{lsb_q} = \mathbf{lsb_{IC}}$ are either equal to 1 or 2, while for the case $\mathbf{lsb_q} = \frac{1}{2}\mathbf{lsb_{IC}}$ the majority of the coefficients are equal to 1.5.

3.1.1 Implementation of LMS1b truncated multipliers

Let us consider, as an example, a signed multiplier with $n = 8$ and $h = 2$, (as in Fig. 3.1). By using the data in Tab. 3.1, the compensation function in this

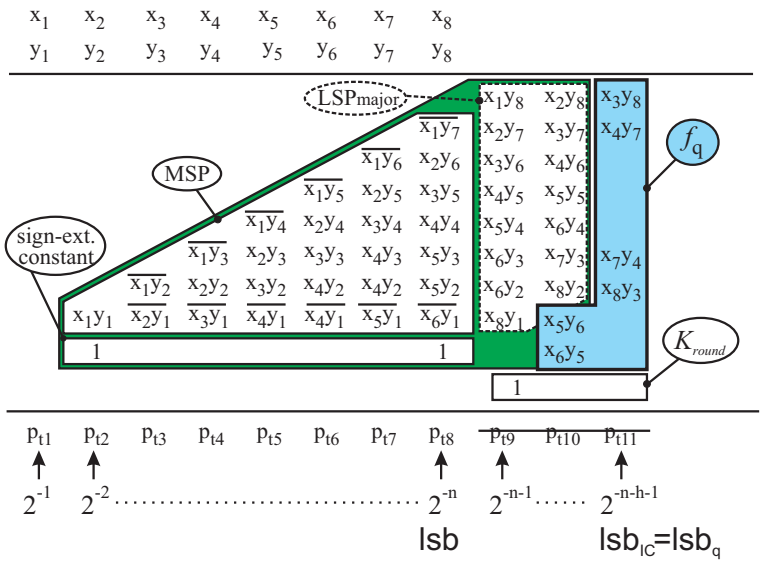

Figure 3.2: Implementation of the proposed (signed) LMS1b truncated multiplier, $\mathbf{lsb_q} = \mathbf{lsb_{IC}}$, $n = 8$ and $h = 2$.

case can be written as:

$$f_q(\text{IC}) = \mathbf{lsb_{IC}} \cdot [\gamma_1 + \gamma_2 + 2\gamma_3 + 2\gamma_4 + \gamma_5 + \gamma_6] \tag{3.1}$$

Thus, to sum $f_q(\text{IC})$ we simply put the terms $\gamma_1, \gamma_2, \gamma_5, \gamma_6$ on the matrix column with weight $\mathbf{lsb_{IC}}$, while the terms γ_3 and γ_4 are aligned on the next matrix column (on the left), having a weight $2\mathbf{lsb_{IC}}$. The corresponding PPM is shown in Fig. 3.2. In general, the PPM of the LMS1b truncated multiplier leaves the two extreme couples of IC partial products (γ_1, γ_2 and $\gamma_{n_{eq}-1}, \gamma_{n_{eq}}$) on the IC column, while the remaining partial products of the IC, having $\gamma_i = 2$, are placed in the column at the left of the IC. The summation of Fig. 3.2 is a "conventional" PPM that can be efficiently implemented using the TDM technique followed by a carry-propagate Adder.

Tab. 3.2 presents the implementation results of the LMS1b truncated multiplier implemented in a $0.18\mu m$ technology. For each row the results report the TDM reduction method ([5] or [7]) giving better performances. The LMS1b truncated multiplier provides a significant area, power and delay improvement with respect to the full-rounded multiplier. As an example for the

Table 3.2: Comparison of the performance of proposed truncated (signed) multipliers. Circuits are implemented in TSMC $0.18\mu m$ technology.

n, h	Multiplier	Reduction Tree			Performances		
		Ref.	#FA	#HA	Area $[10^3 \cdot \mu m^2]$	Power $\left[\frac{\mu W}{MHz}\right]$	Delay [ns]
8, -	full rounded	[5]	37	5	4.51	8.24	2.10
8, 2	LMS1b truncated	[5]	31	1	3.38	6.67	2.10
	LMS2b truncated	[5]	35	1	3.62	7.12	2.10
	LMS2b truncated, IM1	[5]	32	2	3.73	6.97	2.10
	LMS2b truncated, IM2	[7], [5]	33	1	3.62	7.12	2.10
16,-	full rounded	[5]	197	13	19.6	37.9	2.75
16,1	LMS1b truncated	[5]	119	1	10.7	22.2	2.75
	LMS2b truncated	[7]	135	0	11.8	24.5	2.75
	LMS2b truncated, IM1	[5]	125	2	11.8	24.2	2.75
	LMS2b truncated, IM2	[7], [5]	122	2	11.3	23.8	2.75
24,-	full rounded	[5]	485	21	45.2	87.1	3.13
24,1	LMS1b truncated	[5]	275	1	23.8	49.1	3.09
	LMS2b truncated	[7]	299	0	26.0	51.9	3.10
	LMS2b truncated, IM1	[5]	285	2	25.1	51.2	3.10
	LMS2b truncated, IM2	[7], [5]	279	2	24.1	50.4	3.10

case $n = 24, h = 1$ the LMS1b truncated multiplier provides a power and area reduction of 47% and 42%, respectively, while being also slightly faster than the full-rounded multiplier. The good performance are due to the considered compensation function that is well suited to TDM implementation without the need of additional logic. As it will be shown in Sec. 3.2, this is not true for other truncated multipliers proposed in the Literature (see, for instance, [13]) that may require slow and power hungry ripple-like logic to calculate the com-

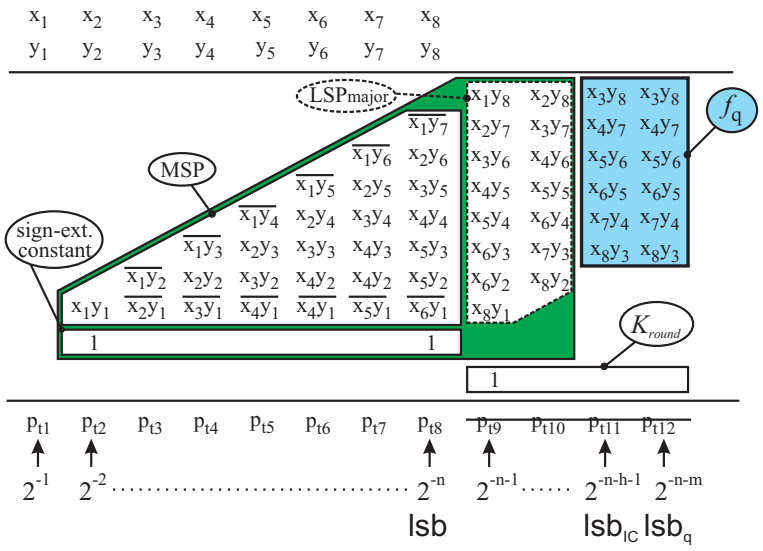

Figure 3.3: Straightforward implementation of the proposed signed LMS2b truncated multiplier $\mathbf{lsb_q} = \frac{1}{2}\mathbf{lsb_{IC}}$, $n = 8$ and $h = 2$.

pensation function.

3.1.2 Implementations of LMS2b truncated multipliers

Let us consider the same signed multiplier example with $n = 8$ and $h = 2$ ($n_{eq} = n - h = 6$). Since REM$(n_{eq}, 4) = 2$, according to Tab. 3.1, every q_i is equal to 1.5. Therefore, the compensation function can be written as:

$$f_q(\text{IC}) = \frac{1}{2}\mathbf{lsb_{IC}} \cdot [3\gamma_1 + 3\gamma_2 + 3\gamma_3 + 3\gamma_4 + 3\gamma_5 + 3\gamma_6] \qquad (3.2)$$

Thus, to sum $f_q(\text{IC})$ we can replicate all the terms $\gamma_1, \gamma_2, \ldots, \gamma_6$ on the two matrix columns with weight $\mathbf{lsb_{IC}}$ and $\mathbf{lsb_{IC}}/2$. This gives the PPM shown in Fig. 3.3. As it can be seen, in Fig. 3.3 each partial product of the IC is inserted twice in the multiplier matrix.

The implementation results to be a bit different if we consider a different configuration, for example $n = 8, h = 3$ ($n_{eq} = n - h = 5$), for which REM$(n_{eq}, 4) = 1$. Hence, from Tab. 3.1, the compensation function can be

written as:

$$f_q(\text{IC}) = \mathbf{lsb_{IC}} \cdot \gamma_1 + \frac{1}{2}\mathbf{lsb_{IC}} \cdot [3\gamma_2 + 3\gamma_3 + 3\gamma_4 + 3\gamma_5] \qquad (3.3)$$

Thus, to sum $f_q(\text{IC})$, γ_1 can be put on the matrix column with weight $\mathbf{lsb_{IC}}$, while the remaining terms $\gamma_2, \ldots, \gamma_6$ are replicated on the two matrix columns with weight $\mathbf{lsb_{IC}}$ and $\mathbf{lsb_{IC}}/2$. This gives the PPM shown in Fig. 3.4.

In general, from Tab. 3.1, in LMS2b truncated multipliers the partial products of the IC can be divided in two subsets. The terms with $q_i = 0.5$ or $q_i = 1$, are inserted only once in the truncated multiplier matrix (either in the IC column or in the column at the right of the IC). The terms with $q_i = 1.5$, are duplicated and inserted both in the IC column and in the column at the right of the IC. The presence of partial products that are inserted twice in the matrix results in an increase in hardware complexity with respect to the LMS1b truncated multiplier. On the other hand the resulting PPM is not a conventional one, since some partial products are duplicated. It is therefore expected that standard TDM implementation strategies will not provide optimal performances for this circuit since they do not exploit the redundancy of the PPM.

This is confirmed in Tab. 3.2 where the performances of the LMS2b truncated multiplier implemented using the standard TDM of [5] and [7] to reduce the PP matrix, and a final Kogge-Stone adder [22], to compute the result are shown. As it can be seen, using the standard TDM results in a noticeable increase in circuit complexity and power dissipation with respect to LMS1b truncated multipliers. For the case $n = 8$, $h = 2$, the matrix of the LMS1b truncated multiplier (shown in Fig. 3.2) is reduced with 31 full-adders (FA) and 1 half-adder (HA). The matrix of the LMS2b truncated multiplier (shown in Fig. 3.3) needs 35 FA and 1 HA with a 7% increase of both area and power dissipation with respect to the LMS1b truncated multiplier. An even worse behavior is observed for larger *n* values (16 and 24 bit multipliers in Tab. 3.2), where the area and power increase of LMS2b truncated multipliers with respect to LMS1b truncated multipliers is up to 10%.

In the following two approaches will be proposed, Implementation Method 1 (IM1) and Implementation Method 2 (IM2), with which the complexity of the LMS2b truncated multipliers is reduced by exploiting the redundancy of the PPM.

Implementation Method 1 Let us name γ_i' the PPs of the IC with $q_i = 1.5$. In the implementation approach IM1, the PPs γ_i' are grouped in couples

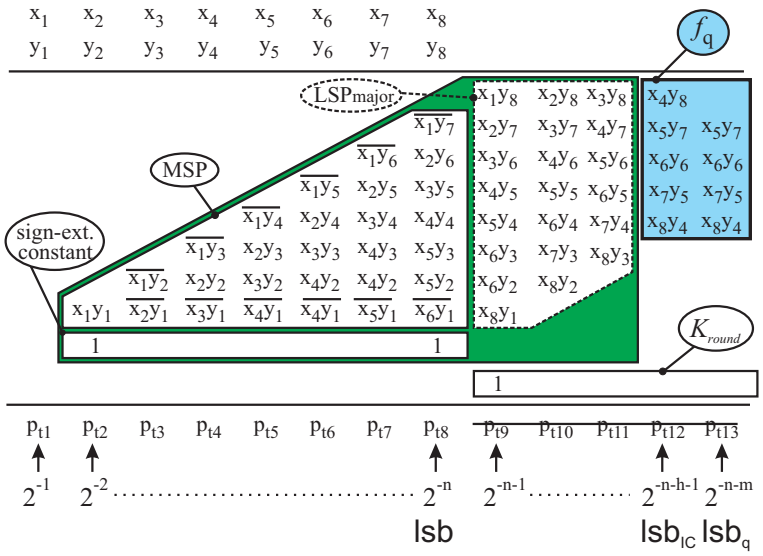

Figure 3.4: Straightforward implementation of the proposed signed LMS2b truncated multiplier $\mathbf{lsb_q} = \frac{1}{2}\mathbf{lsb_{IC}}$, $n = 8$ and $h = 3$.

$\Gamma_i = (\gamma'_{2i-1}, \gamma'_{2i})$. The contribution of the couple Γ_i to the total summation is:

$$\Delta_i = \frac{1}{2}\mathbf{lsb_{IC}} \cdot \left[3\gamma'_{2i-1} + 3\gamma'_{2i}\right] \tag{3.4}$$

In the above equation the partial products γ'_{2i-1} and γ'_{2i} are either 0 or 1. Therefore $\Delta_i \in \{0,\ 1.5\mathbf{lsb_{IC}},\ 3\mathbf{lsb_{IC}}\}$ and can be expressed as follows:

$$\Delta_i = \frac{1}{2}\mathbf{lsb_{IC}} \cdot \left(\eta 2_i \cdot 2^2 + \eta 1_i \cdot 2^1 + \eta 0_i \cdot 2^0\right) \tag{3.5}$$

where $\eta 2_i, \eta 1_i$ and $\eta 0_i$ are three binary values. By using (3.5) three terms are introduced in the PPM to account Δ_i, while four partial products are needed in the simple approach that replicates the partial products on the two matrix columns with weight $\mathbf{lsb_{IC}}$ and $\mathbf{lsb_{IC}}/2$. The relation between $(\gamma'_{2i-1}, \gamma'_{2i})$ and $(\eta 2_i, \eta 1_i, \eta 0_i)$ is shown in Tab. 3.3 and is given by:

$$\begin{aligned} \eta 2_i &= \gamma'_{2i-1} \wedge \gamma'_{2i} \\ \eta 1_i &= \gamma'_{2i-1} \vee \gamma'_{2i} \\ \eta 0_i &= \gamma'_{2i-1} \oplus \gamma'_{2i} \end{aligned} \tag{3.6}$$

Table 3.3: Truth table describing the Boolean relationship between $(\gamma'_{2i-1}, \gamma'_{2i})$ and $(\eta 2_i, \eta 1_i, \eta 0_i)$.

γ'_{2i-1}	γ'_{2i}	$\Delta_i/\mathbf{lsb_{IC}}$	$\eta 2_i$	$\eta 1_i$	$\eta 0_i$
0	0	0	0	0	0
0	1	1.5	0	1	1
1	0	1.5	0	1	1
1	1	3	1	1	0

Fig. 3.5 shows the implementation of the LMS2b truncated multiplier with $n = 8$, $h = 2$ using IM1 approach. The six terms of the IC $(x_3y_8, x_4y_7, \ldots, x_8y_3)$ are grouped in three couples $(\Gamma_1, \Gamma_2, \Gamma_3)$ that feed the ηi_j *generation logic* that, using (3.6), computes the ηi_j terms. These terms are inserted in the PP matrix of the multiplier in order to compute $f_q(\text{IC})$. Comparing the IM1 circuit of Fig. 3.5 with the standard TDM of Fig. 3.3, it is evident that the number of terms needed to compute $f_q(\text{IC})$ reduces from 12 to 9 at the expense of a small complexity increase due to ηi_j *generation logic*.

A slightly different implementation is required in other cases. Fig. 3.6 shows the IM1 circuit for $n = 8$, $h = 3$. Since $n_{eq} = 5$, $\gamma_1 = x_4y_8$ has $q_1 = 1$ and is directly inserted in the PP matrix. The remaining four terms of the IC $(x_5y_7, \ldots, x_8y_4)$ have $q_i = 1.5$. These PPs form two couples Γ_1, Γ_2, which produce (through the ηi_j *generation logic*) six terms in the PP matrix.

Tab. 3.2 shows the implementation data of the IM1 LMS2b truncated multipliers. Note that the LMS2b truncated multiplier of Fig. 3.5 is reduced by a carry-save tree with 32 FA and 2 HA, that is, with only one FA and one HA in addition with respect to the circuit of Fig. 3.2. Comparing the standard TDM and IM1 in Fig. 3.6 it is evident that the latter provides better performance as n increases. As an example, for $n = 24$, IM1 multiplier is 4% smaller than conventional implementation and only the 5% larger than the LMS1b truncated multiplier. For $n = 8$ the IM1 yields a slightly larger circuit with more power consumption with respect to the standard TDM. In this case, the reduction in carry-save tree complexity is nullified by the additional logic required to compute ηi_j terms.

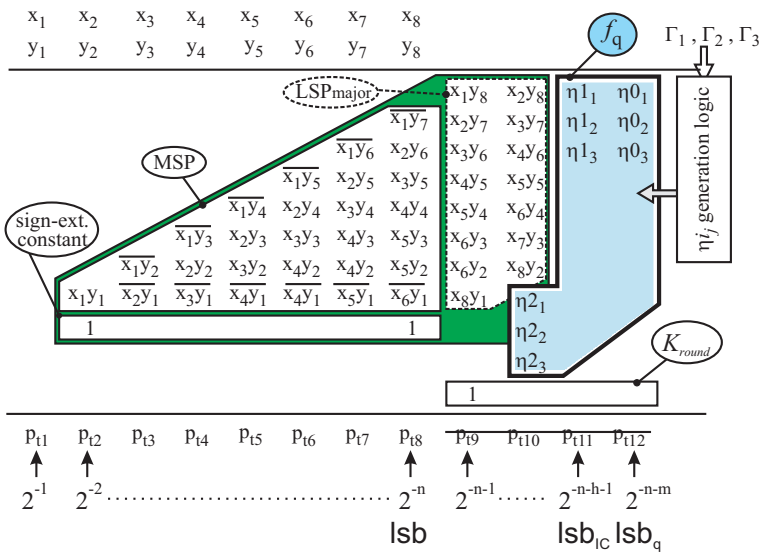

Figure 3.5: Implementation Method 1 (IM1) of the proposed (signed) LMS2b truncated multiplier $\mathbf{lsb_q} = \frac{1}{2}\mathbf{lsb_{IC}}$, $n = 8, h = 2$

Implementation Method 2 This improved implementation strategy for LMS2b truncated multipliers fully exploits the hardware sharing between the columns of the PPM.

Fig. 3.7 presents the IM2 approach for the LMS2b truncated multiplier with $n = 8$, $h = 2$. In this case, as shown in (3.2), all the six q_i are equal to 1.5. An auxiliary carry-save reduction tree sums the six IC terms. In the small example of Fig. 3.7, the auxiliary carry save reduction tree is composed by two full adders, whose carry and sum output are indicated as c_1, c_0 and s_1, s_0 respectively. Let us indicate as S and C the results of the auxiliary carry-save tree: $S = s_1 \cdot 2^1 + s_0 \cdot 2^0$; $C = c_1 \cdot 2^1 + c_0 \cdot 2^0$. We have:

$$S + C = \gamma_1 + \gamma_2 + \gamma_3 + \gamma_4 + \gamma_5 + \gamma_6 \tag{3.7}$$

Therefore, the value of $f_q(\text{IC})$ in (3.2) can be expressed as:

$$f_q(\text{IC}) = \frac{1}{2}\mathbf{lsb_{IC}} \cdot (S + C) + \mathbf{lsb_{IC}} \cdot (S + C) \tag{3.8}$$

According to (3.8), the bits s_i and c_i are inserted twice in the final PPM of the truncated multiplier. Starting from the leftmost column (with weight

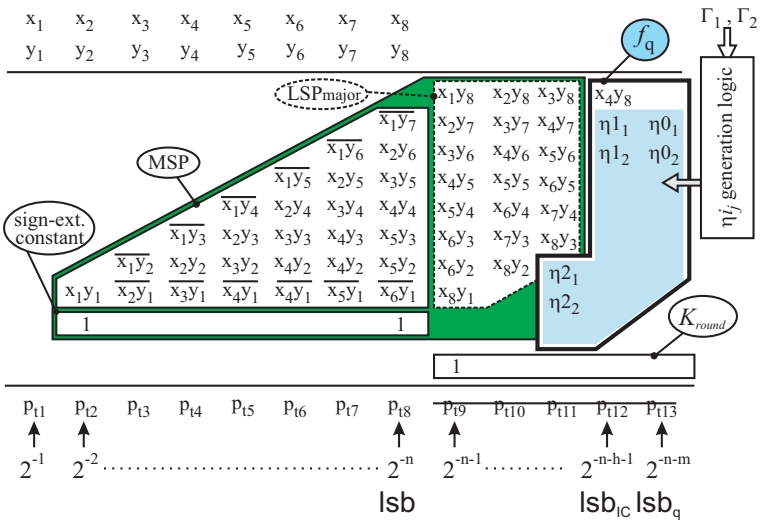

Figure 3.6: Implementation Method 1 (IM1) of the proposed (signed) LMS2b truncated multiplier $\mathbf{lsb_q} = \frac{1}{2}\mathbf{lsb_{IC}}$, $n = 8, h = 3$

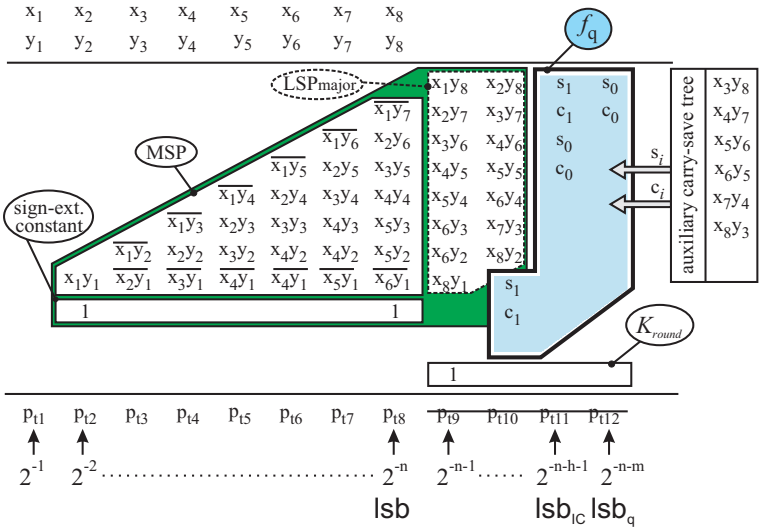

Figure 3.7: Implementation Method 2 (IM2) of the proposed (signed) LMS2b truncated multiplier $\mathbf{lsb_q} = \frac{1}{2}\mathbf{lsb_{IC}}$, $n = 8, h = 2$

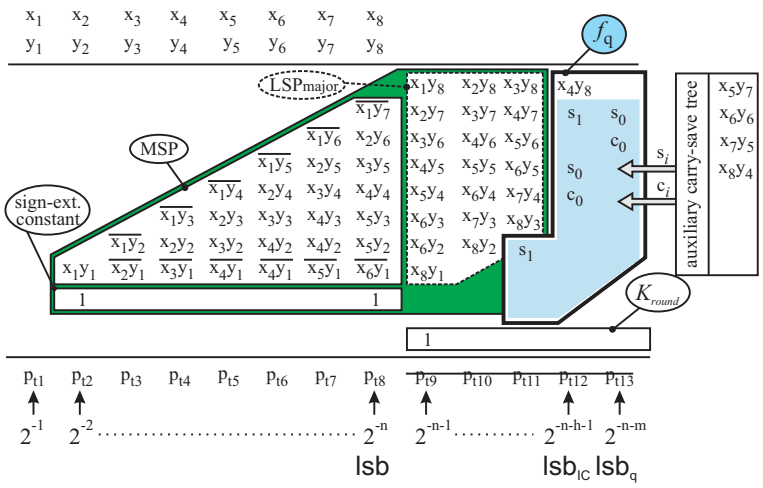

Figure 3.8: Implementation Method 1 (IM1) of the proposed (signed) LMS2b truncated multiplier $\mathbf{lsb_q} = \frac{1}{2}\mathbf{lsb_{IC}}$, $n = 8, h = 3$

$1/2\mathbf{lsb_{IC}}$), each bit s_i and c_i is inserted once in the i^{th} column and once in the $(i+1)^{th}$ column of the matrix. This is shown in Fig. 3.7 for the example $n = 8$, $h = 2$. As it can be seen, the IM2 approach in this example introduces 8 terms in the matrix to compute $f_q(\mathrm{IC})$. This compares favorably with the 9 terms needed by the IM1 approach (see Fig. 3.5) and the 12 terms needed by the standard TDM.

Fig. 3.8 shows the IM2 technique applied to the LMS2b truncated multiplier with $n = 8$, $h = 3$. In this case the four terms with $q_i = 1.5$ (see (3.3)), are compressed by using the auxiliary carry-save reduction tree, that includes a single full-adder. Note that $c_1 = 0$ and is not introduced in Fig. 3.8. The term $\gamma_1 = x_4 y_8$ has $q_i = 1$ and is hence directly inserted in the matrix column with weight $\mathbf{lsb_{IC}}$ of the truncated multiplier. In the case of Fig. 3.8, we have a total of seven terms that are included in the PPM to compute $f_q(\mathrm{IC})$.

In general, the wordlengths at the output of the auxiliary carry-save tree (i.e. the wordlenghts of *S* and *C* signals) increase logarithmically with n_{eq}. As a consequence, the number of terms to be included in the PPM to compute $f_q(\mathrm{IC})$ also increases logarithmically with n_{eq}. On the contrary, the number of ηi_j terms needed using the IM1 approach increases linearly with n_{eq}. Therefore the IM2 approach is more and more effective as n_{eq} increases. It

is worthwhile to note that TDM is able to compensate input delay asymmetries. Therefore, the delays of the auxiliary carry-save reduction tree are partly absorbed by the TDM used to reduce the multiplier PPM, with a small delay penalty.

Implementation data of the IM2 LMS2b truncated multipliers are shown in Tab. 3.2. As it can be observed, the best performance are obtained by using [7] for the auxiliary carry-save reduction tree, while the TDM approach of [5] resulted, always, the best technique to compress the final matrix. The IM2 LMS2b truncated multipliers provide the best performances in every considered case. For $n = 16$, $h = 1$, the standard TDM LMS2b truncated multiplier results in 10% area increase with respect to the LMS1b truncated multiplier. Using the IM2 method, the area increase with respect to LMS1b truncated multiplier is only 5%.

3.2 Truncated Multipliers Performances

This section is devoted to the comparison between LMS1b and LMS2b truncated multipliers proposed in this chapter and previously proposed architectures. From the result of the previous section, LMS2b truncated multipliers will be implemented by using the IM2 approach. In the comparison with literature results, the attention is restricted to variable-correction truncated multipliers [10, 11, 12, 13, 14, 15, 16, 17, 19] which provide much lower errors with respect to constant-correction circuits (see Ch. 1).

It is worth to highlight that not all the above cited papers discuss the implementation of truncated multipliers with $h > 0$ and deal with both signed and unsigned cases. A review of the variable-correction approaches found in the Literature from this point of view is reported in Tab. 3.4.

In [13] Jou *et al.* consider the case of both signed and unsigned multipliers. In [13] truncated multipliers with additional w columns in the matrix are considered, however the number of output bits of the multiplier in [13] is not equal to n but it is instead $n + w$. In this thesis (see Fig. 3.1), similarly to the rest of the Literature, the number of output bits is fixed[1] (is equal to n) and h is a design parameter that can be used to increase the accuracy without changing the weight of the output **lsb**. Therefore the architecture of [13] can be compared with the proposed multipliers only when $h = 0$. However, as shown in Ch. 2, the results obtained for unsigned multiplier with $h = 0$ can be extended rather

[1] In is worth highlighting that proposed approaches can be easily extended to consider cases where the number of output bits is larger than n, still keeping h as an accuracy parameters.

Table 3.4: Previously proposed truncated multipliers considered in the comparison.

Truncated multiplier	Signed h=0	Unsigned h=0	Signed/Unsigned h>0
Jou *et al.* [13]	proposed in the original paper	proposed in the original paper	can be extended to this case
Van *et al.* [15]-[16]	proposed in the original paper	can be extended to this case	proposed in the original paper
Curticapean *et al.* [14]	-	proposed in the original paper	can be extended to this case
Kuang *et al.* [18]	-	proposed in the original paper	can be extended to this case
Liao *et al.* [17]	proposed in the original paper	-	-
Swartzlander *et al.* [11, 10, 23]	-	proposed in the original paper	proposed in the original paper

straightforwardly to cover both signed and unsigned multipliers with $h > 0$. Thus, as highlighted in Tab. 3.4, the approach of [13] has been extended to the case $h > 0$, by introduction of a suitable rounding constant which has been evaluated in order to minimize the total mean error (as discussed in Ch. 2, this also minimizes the total mean square error).

The same considerations apply to the multipliers proposed by Kuang *et al.* [18]. In this case, however, the original paper considers only unsigned multiplier. Therefore the technique of [18] cannot be extended to signed multipliers with $h = 0$. Please note that [18] proposes two approaches. It will be considered only the first approach (Type *I*) that provides a lower mean square error.

The truncated multipliers proposed by Van *et al.* [15]-[16] include signed multipliers for $h \geq 0$. The error compensation function proposed for $h > 0$ can however be extended to the unsigned multiplier with $h = 0$ (similarly to what has been done for [13],[18]).

Curticapean *et al.* [14] consider only the unsigned multiplier with $h = 0$. The architecture proposed in [14] can hence be extended (still adding a suitable

rounding constant) to signed and unsigned multipliers with $h > 0$.

The variable-correction multipliers proposed by Swartzlander *et al.* [11, 10, 23] consider the unsigned case for $k \geq 0$. This technique, therefore, cannot be extended to the signed multipliers with $h = 0$.

Finally the approach of Liao *et al.* [17] is originally developed for the signed $h = 0$ case and cannot be extended to any other case, so it will be not considered in the following.

In the following the multipliers will be compared in term of mean square error (Sec. 3.2.1), maximum absolute error (Sec. 3.2.2) and in terms of area, power and propagation delay (Sec. 3.2.3).

3.2.1 Mean Square Error Performances

The Tables 3.5-3.6 compare the mean square error obtained by the LMS1b and LMS2b truncated multipliers proposed and the state of art architectures. The first two columns in those Tables report the error achieved by the truncated multiplier proposed, as obtained with the analytical formulas presented in Ch. 2. The remaining columns in Tabs. 3.5-3.6 show, instead, the mean square error as obtained after exhaustive simulation. Since the simulation time increases as $O(2^{2n})$, an unreasonable amount of CPU time is needed when n is larger than 16. For this reason, in Tabs. 3.5-3.6, simulation data are not available for $n = 32$ and for $n = 64$.

The data in Tabs. 3.5-3.6 highlight a very good agreement between theory and simulations for the proposed multipliers. The theoretical values of LMS1b truncated multipliers are exact for $h = 0$, remaining data are almost exact.

LMS1b truncated multipliers exhibit good error performance. When $h = 0$ only the multiplier of Kuang *et al.* [18] is able to obtain a slightly lower error than the LMS1b truncated multiplier. On the other hand, the multiplier of Kuang *et al.* [18] yields a larger error than LMS1b truncated multiplier for $h > 0$. In a few cases, when $h = 1$, LMS1b truncated multipliers presents also a slightly larger error with respect to the multipliers of Van *et al.* [15]-[16] and Jou *et al.* [13]. This happens for $n = 8, 10, 12, 14$. Note that only in few cases LMS1b truncated multiplier presents a slightly larger error, because the better results provided by [18], [15]-[16] and [13] are obtained through exhaustive research, so it isn't possible to find an unique methodology always valid.

LMS2b truncated multiplier provides the lowest error for any *n* and *h* value. As an example in the case $n = 16$, $h = 1$ the LMS2b truncated multiplier results in a reduction of mean square error of 13%, 13%, 39%, 31%, and 16%

Table 3.5: Theoretical and simulated mean square errors of proposed and previously proposed truncated multiplier (S=signed multiplier; U=unsigned multiplier), $n = 8, 10, 12$ and $h = 0, 1, 2, 3$.

n	h	Theoretical $\varepsilon^2_{\text{total}}(\mathbf{lsb}^2)$		Simulated Mean Square Error $\varepsilon^2_{\text{total}}(\mathbf{lsb}^2)$											
		LMS1b	LMS2b	LMS1b	LMS2b	Jou [13]	Van [15]-[16]		Curticapean [14]		Kuang I [18]		Swartzlander [11, 10, 23]	
		(S/U)	(S/U)	(S/U)	(S/U)	(S/U)	(S)	(U)	(S)	(U)	(S)	(U)	(S)	(U)
8	0	0.216	0.190	0.216	0.190	0.598	0.263	-	-	0.234	-	0.213	-	0.263
	1	0.118	0.105	0.118	0.105	0.104	0.104		0.187		0.159		0.123	
	2	0.091	0.087	0.090	0.087	0.087	0.087		0.108		0.101		0.092	
	3	0.085	0.084	0.085	0.084	0.084	0.084		0.089		0.085		0.085	
10	0	0.258	0.227	0.258	0.227	0.700	0.305	-	-	0.285	-	0.255	-	0.305
	1	0.130	0.114	0.129	0.114	0.119	0.119		0.196		0.170		0.134	
	2	0.094	0.090	0.094	0.090	0.090	0.090		0.110		0.104		0.095	
	3	0.086	0.085	0.086	0.085	0.085	0.085		0.090		0.088		0.086	
12	0	0.300	0.257	0.300	0.257	0.780	0.347	-	-	0.333	-	0.296	-	0.347
	1	0.140	0.121	0.140	0.121	0.134	0.134		0.206		0.180		0.144	
	2	0.096	0.091	0.096	0.091	0.094	0.094		0.113		0.106		0.097	
	3	0.086	0.085	0.086	0.085	0.086	0.086		0.090		0.089		0.086	

Table 3.6: Theoretical and simulated mean square errors of proposed and previously proposed truncated multiplier (S=signed multiplier; U=unsigned multiplier) $n = 14, 16, 32, 64$ and $h = 0, 1, 2, 3$.

n	h	Theoretical $\varepsilon^2_{\text{total}}(\mathbf{lsb}^2)$		Simulated Mean Square Error $\varepsilon^2_{\text{total}}(\mathbf{lsb}^2)$											
		LMS1b (S/U)	LMS2b (S/U)	LMS1b (S/U)	LMS2b (S/U)	Jou [13] (S/U)	Van [15]-[16] (S)	(U)	Curticapean [14] (S)	(U)	Kuang I [18] (S)	(U)	Swartzlander [11, 10, 23] (S)	(U)
14	0	0.342	0.290	0.342	0.290	0.847	0.389	-	-	0.380	-	0.338	-	0.389
	1	0.151	0.131	0.151	0.131	0.147	0.147		0.216		0.191		0.155	
	2	0.099	0.095	0.099	0.095	0.098	0.098		0.115		0.109		0.100	
	3	0.087	0.086	0.087	0.086	0.087	0.087		0.091		0.089		0.087	
16	0	0.384	0.326	0.384	0.326	0.905	0.431	-	-	0.425	-	0.380	-	0.431
	1	0.161	0.139	0.160	0.139	0.160	0.160		0.227		0.201		0.165	
	2	0.101	0.096	0.102	0.096	0.101	0.101		0.118		0.112		0.102	
	3	0.088	0.086	0.088	0.086	0.087	0.087		0.092		0.090		0.088	
32	0	0.717	0.600	N.A.	N.A.	N.A.	N.A.	-	-	N.A.	-	N.A.	-	N.A.
	2	0.122	0.113	N.A.	N.A.	N.A.	N.A.		N.A.		N.A.		N.A.	
	4	0.086	0.085	N.A.	N.A.	N.A.	N.A.		N.A.		N.A.		N.A.	
64	0	1.384	1.141	N.A.	N.A.	N.A.	N.A.	-	-	N.A.	-	N.A.	-	N.A.
	2	0.164	0.147	N.A.	N.A.	N.A.	N.A.		N.A.		N.A.		N.A.	
	4	0.088	0.087	N.A.	N.A.	N.A.	N.A.		N.A.		N.A.		N.A.	

with respect to [13], [15]-[16], [14], [18], and [11, 10, 23], respectively. Even larger improvements are achieved for $h = 0$.

The availability of analytical formulas allows us to evaluate the mean square error of the proposed multipliers also for large n values, while this is not possible for previously proposed architectures. Data in Tab. 3.5-3.6 show that, by using $h = 4$, the proposed truncated multipliers yield an error very close to full-rounded multipliers ($\mathbf{lsb}^2/12 = 0.083 \cdot \mathbf{lsb}^2$), also for very large n values (e.g. $n = 64$).

3.2.2 Maximum Absolute Error Performances

The proposed analytical formula for the maximum error is compared against numerical simulations of the LMS1b truncated multipliers with different bit width. Furthermore, the maximum absolute error performances of the LMS1b truncated multiplier are compared with the maximum absolute error of state of art truncated multipliers and LMS2b multiplier. The results are shown in Table 3.7.

The comparison of the first two columns shows that the analytical formula, eq. (2.126) is in perfect agreement with the simulated results, demonstrating the correctness of the analytical formula devised in the previous chapter.

The comparison of the maximum absolute error performances highlights the fact that the LMS1b truncated multiplier provides good maximum absolute error performances if compared with the papers presented in the literature. For $h = 0$ the only reference that overcomes the LMS1b truncated multiplier is [14]. For $h \neq 0$ also [13] and [15]-[16] give a slightly lower maximum absolute error.

But the important point is that LMS1b multiplier is the only one multiplier with an analytical expression of the maximum absolute error. Hence when the number of bits is higher than 16, the only available results are those obtained with the analytical formula calculated in Ch. 2. In fact, obtaining the same results with a brute force simulation of the truncated multiplier is impossible due to the huge amount of needed CPU time. Recalling what has been told in Ch. 2, the maximum absolute error of a truncated multiplier is only achieved for a very small number of input values, hence non-exhaustive simulations are very unlikely to provide the actual maximum absolute error value or an error value near to the maximum.

Table 3.7: Theoretical and simulated maximum absolute error of proposed and previously proposed truncated multipliers (S=signed multiplier; U=unsigned multiplier), $n = 10, 12, 14, 20, 24, 32, 64$ and $h = 0, 1, 2, 3$.

n	h	Theoretical $\varepsilon_{\max}(\mathbf{lsb})$	Simulated Mean Square Error $\varepsilon_{\max}(\mathbf{lsb})$											
		LMS1b	LMS1b	LMS2b	Jou [13]	Van [15]-[16]		Curticapean [14]		Kuang I [18]		Swartzlander [11, 10, 23]	
		(S/U)	(S/U)	(S/U)	(S/U)	(S)	(U)	(S)	(U)	(S)	(U)	(S)	(U)
10	0	2.055	2.055	2.000	2.346	2.055	-	-	1.888	-	2.641	-	2.055
	1	1.172	1.172	1.125	1.090	1.090		1.361		1.277		1.194	
	2	0.805	0.805	0.750	0.752	0.752		0.888		0.847		0.805	
	3	0.625	0.625	0.581	0.606	0.606		0.673		0.652		0.631	
12	0	2.388	2.388	2.500	2.680	2.388	-	-	2.222	-	2.972	-	1.361
	1	1.340	1.340	1.313	1.257	1.257		1.527		1.444		1.361	
	2	0.888	0.888	0.875	0.836	0.836		0.972		0.930		0.888	
	3	0.668	0.668	0.656	0.647	0.647		0.715		0.694		0.673	
14	0	2.722	2.722	3.000	3.013	2.722	-	-	2.555	-	3.305	-	2.722
	1	1.506	1.506	1.625	1.423	1.423		1.694		1.611		1.527	
	2	0.972	0.972	1.000	0.920	0.920		1.055		1.013		0.972	
	3	0.710	0.710	0.703	0.689	0.689		0.756		0.736		0.715	
20	0	3.72	N.A.	N.A.	N.A.	N.A.	-	-	N.A.	-	N.A.	-	N.A.
24	0	4.39	N.A.	N.A.	N.A.	N.A.	-	-	N.A.	-	N.A.	-	N.A.
32	0	5.72	N.A.	N.A.	N.A.	N.A.	-	-	N.A.	-	N.A.	-	N.A.
64	0	11.06	N.A.	N.A.	N.A.	N.A.	-	-	N.A.	-	N.A.	-	N.A.

3.2.3 Electrical Performances (Area Occupation, Power Dissipation, Propagation Delay)

As observed before, the proposed truncated multipliers are well suited for state-of-the-art tree based implementations (including a TDM carry-save tree and a final carry-prefix propagate adder) [24, 5, 7] since the circuit output can be written as a summation matrix of partial products. This remains true for many previously proposed truncated multipliers ([18]-[11, 10, 23]). The papers of Jou *et al.* [13], Van *et al.* [15]-[16] and Curticapean *et al.* [14], on the other hand, consider array multiplier implementations which, because of ripple architecture, are very slow and power hungry. These approaches can be still implemented with a carry-save TDM reduction method. However, it is worth to highlight that the terms added to the PPM (related to the computation of the compensation function) are still obtained with a ripple *AND*/ *OR* network, which increases circuit power dissipation and propagation delay.

In order to have a fair comparison among every approach, the multipliers previously proposed in literature have been implemented with a carry-save TDM reduction tree followed by a fast carry-propagate adder. All the circuits have been implemented in a $0.18\mu m$ technology. Fig. 3.9-3.10 shows the results of the implementations obtained by varying the delay constraint during synthesis, in the case $n = 16$, $h = 1$.

The reported data show that proposed LMS1b truncated and Kuang *et al.* multipliers exhibit very similar performances both in term of area and power. A second set of multipliers which show very similar performance is composed by LMS2b multiplier, Van *et al.* and Swartzlander *et al.* multipliers. The multipliers proposed by Jou *et al.* and Curticapean *et al.* are less effective than other architectures. As we have observed before, this is due to the ripple structure used in the network which implements the compensation function. This not only limits the minimum delay for which the multiplier can be synthesized but also results in a large glitching, which increases the power dissipation. Since, in addition, Jou *et al.* and Curticapean *et al.* multipliers do not exhibit good error performances (see Tab. 3.5,3.6) in the following we will not consider anymore these two architectures.

It is interesting to observe that also the Van *et al.* multiplier includes a ripple error compensation network. In this case, however, this network reduces to a single multiple-input gate[2] which can be implemented with a tree struc-

[2] Please note that the error compensation network proposed by Van et al. reduces to a single multiple-input gate only for $h > 0$. For $h = 0$ the network of Van et al. multiplier is similar to the networks of Jou et al. and Curticapean et al. multipliers.

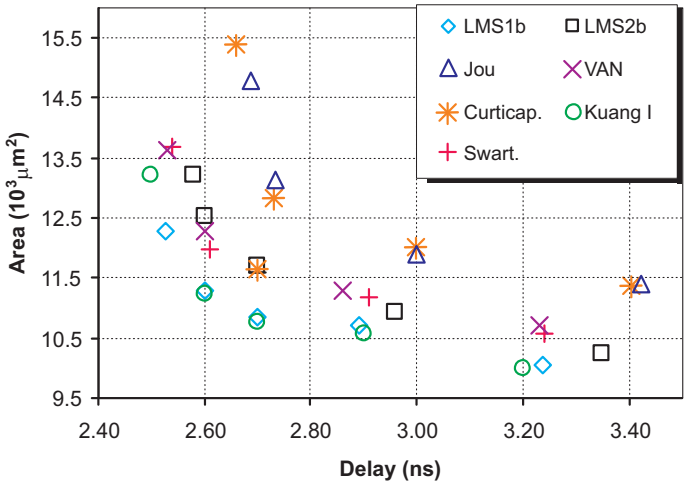

Figure 3.9: Signed Multiplier performances for $n = 16$ and $h = 1$ by varying the delay constrain imposed during circuit synthesis ($0.18\mu m$ technology).

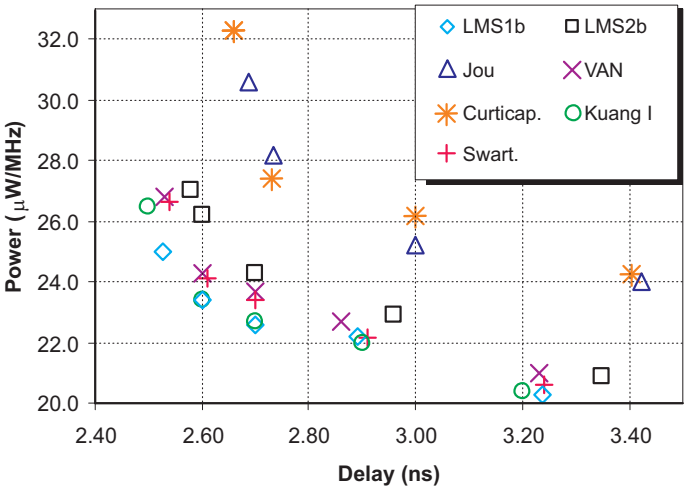

Figure 3.10: Signed Multiplier performances for $n = 16$ and $h = 1$ by varying the delay constrain imposed during circuit synthesis ($0.18\mu m$ technology).

ture. This justifies why Van *et al.* multiplier shows better performances than Jou *et al.* and Curticapean *et al.* approaches.

As shown in Figs. 3.9-3.10, the multiplier of Kuang *et al.* has electrical performance similar to LMS1b truncated multiplier. However the accuracy of Kuang *et al.* multiplier is comparable to LMS1b truncated multiplier only for $h = 0$, while for $h > 0$ the Kuang *et al.* multiplier yields a larger error. As a consequence, the multiplier of Kuang *et al.* has also been excluded in subsequent analysis.

3.2.4 Area versus Accuracy Trade-off

Previous sections analyzed separately the accuracy and the complexity (that is, the silicon area) of the different multipliers. In a real application the accuracy is generally constrained by system-level consideration and we would like to achieve the lowest complexity for a required accuracy.

Parameter *h* is a trade-off parameter between accuracy and complexity that can be used to tune the multiplier accuracy to the requirement imposed by the specific application. A comparison of the different truncated multipliers considering this trade-off between accuracy and complexity is therefore necessary. To that purpose, LMS1b and LMS2b truncated multipliers and the multipliers proposed by Van *et al.*, Liao *et al.* and Swartzlander *et al.* have been simulated and implemented. Figs. 3.11-3.12-3.13-3.14 show the results on a diagram area vs. mean square error for $n = 8, 12, 16$ and 24 bit and $h = 0, 1, 2, 3$. Each diagram, for each value of *h*, shows also the lowest theoretical mean square error achievable with a variable-correction truncated multiplier (the term $\varepsilon^2_{\text{low\_bound}}$ discussed in Ch. 2).

The data of Figs. 3.11-3.12-3.13-3.14 show that for $h = 0$ the proposed LMS1b and LMS2b truncated multipliers exhibit the best trade-off between area and accuracy. The LMS1b truncated multiplier results in the lowest area, while the LMS2b truncated multiplier achieves an error very close to the theoretical limit. The multiplier of Van *et al.* [15]-[16] appears ineffective for $h = 0$ because of the large error in comparison to other multipliers. The data for $h = 1$ in Fig. 3.11-3.12-3.13-3.14 show a picture similar to the case $h = 0$ with the difference that now the multiplier of Van *et al.* is more effective, and competes with LMS2b truncated multiplier.

For $h \geq 2$ the error of all multipliers is very close to the theoretical limit. In turn, for the considered n values, the theoretical limit is very close to the error of the full-rounded multiplier ($1/12 \cdot \mathbf{lsb}^2$). In these cases the best solution is selecting the multiplier with the lowest area, that is generally the proposed

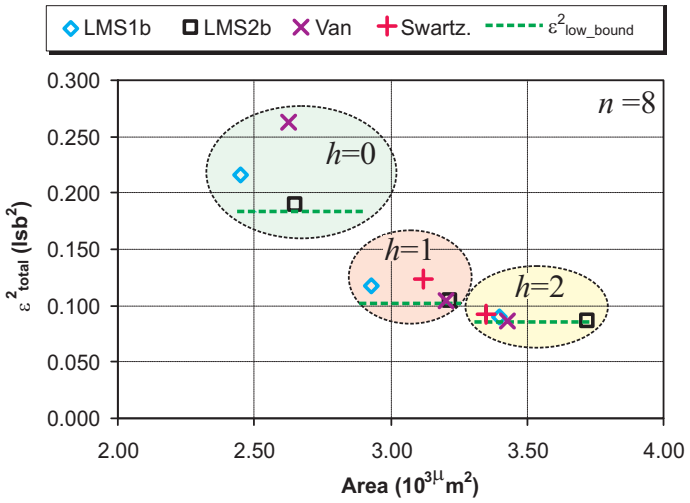

Figure 3.11: Trade off between Area Occupation and Error (mean square total error - $\varepsilon^2_{\text{total}}$) for different Signed Truncated Multipliers, $n = 8$. The mean square error is obtained through simulation.

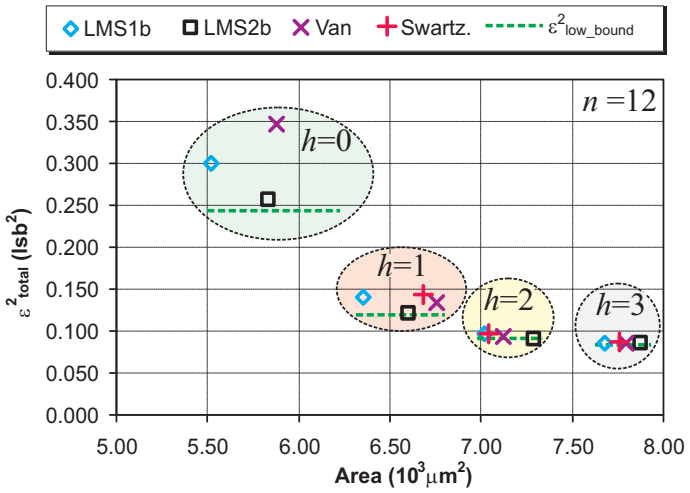

Figure 3.12: Trade off between Area Occupation and Error (mean square total error - $\varepsilon^2_{\text{total}}$) for different Signed Truncated Multipliers, $n = 12$. The mean square error is obtained through simulation.

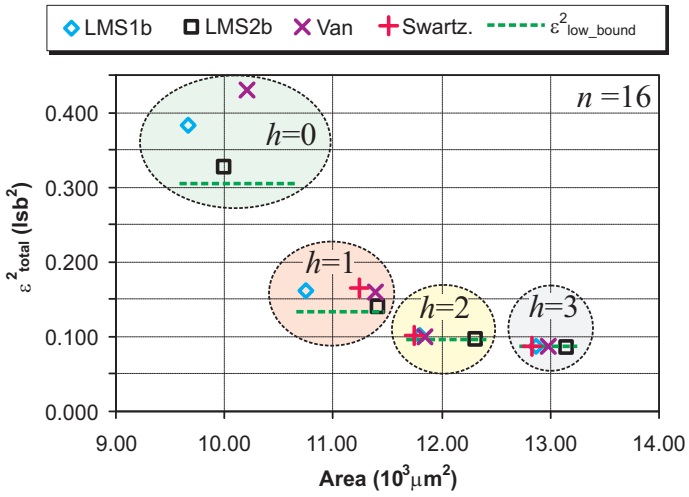

Figure 3.13: Trade off between Area Occupation and Error (mean square total error - $\varepsilon^2_{\text{total}}$) for different Signed Truncated Multipliers, $n = 16$. The mean square error is obtained through simulation.

LMS1b. Note that for $n = 24$ the mean square error cannot be simulated. Therefore the data in Fig. 3.11-3.12-3.13-3.14 for $n = 24$ include only LMS1b and LMS2b truncated multipliers.

3.3 Experimental Verification

The performance of the proposed truncated multiplier have been experimentally verified on a test chip [25]. In this chip it has been implemented a truncated multiplier designed with LMS1b technique for $n = 16$, $h = 0$ and also, as a comparison, a full-rounded multiplier. The technology used is $0.18\mu m$ with 6 levels of metal. The photograph of the test chip is shown in Fig. 3.15.

The experimental performance are summarized in Tab. 3.15. The developed LMS1b truncated multiplier is able to halve the power dissipation and to reduce of about 44% the area occupation. In addition the multiplier designed with the proposed technique is 13% faster than the full-rounded multiplier. These performance increase (relative to full-rounded multiplier) are consistent with data in Tab. 3.2. On the other hand, the actual experimental values reported in Tab. 3.15 are significantly worse with respect to the data in Tab. 3.2.

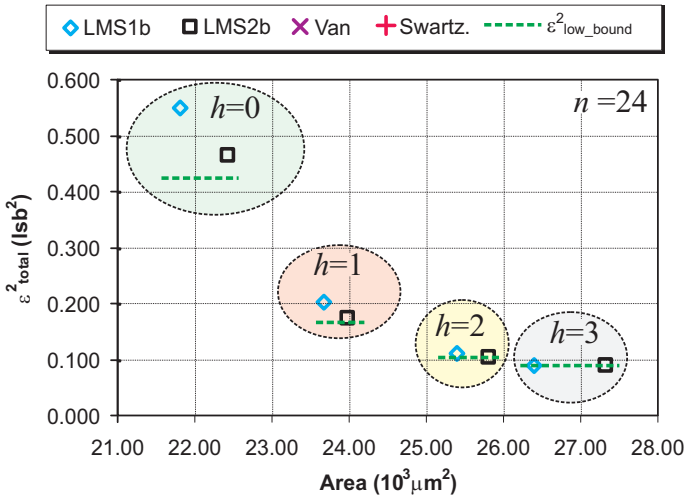

Figure 3.14: Trade off between Area Occupation and Error (mean square total error - $\varepsilon^2_{\text{total}}$) for different Signed Truncated Multipliers, $n = 24$. The mean square error is obtained by the theoretical formulas of Ch. 2.

Table 3.8: Experimental performance of the signed 16 bit multipliers realized in the test chip shown in Fig. 3.15.

Multiplier	f_{clk} $[MHz]$	Area $[10^3 \cdot \mu m^2]$	Power $\left[\frac{\mu W}{MHz}\right]$
LMS1b ($h = 0$)	385	19.1	36.6
full-rounded	340	33.7	73.2

This is due to two reasons. Firstly, an un-optimized standard-cell library (using only Manhattan geometry in the layout) was employed to fabricate the chip, while a more effective vendor-supplied library was considered for the simulations reported in Tab. 3.2. Secondly, data in Tab. 3.2 are synthesis results that do not take into account the overhead due to place and route.

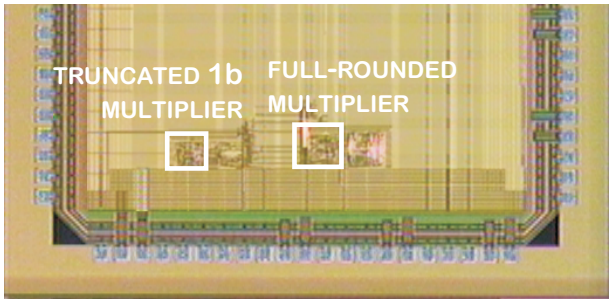

Figure 3.15: Trade off between Area Occupation and Error (mean square total error - $\varepsilon^2_{\text{total}}$) for different Signed Truncated Multipliers. For $n = 16$ the mean square error is simulated. For $n = 24$ the mean square error is obtained by the theoretical formulas of Ch. 2.

3.4 Conclusions

The practical implementation of truncated multipliers with the quantized linear compensation function proposed in Ch. 2 is investigated in this chapter. Efficient implementations, based on a carry-save reduction tree and a final parallel-prefix carry-propagate adder, have been considered.

It has been shown that, when coefficients are quantized with one bit (LMS1b truncated multiplier), the implementation is straightforward. When two bits are employed to quantize coefficients of compensation function (LMS2b truncated multipliers), the optimal implementation is more challenging. The best approach uses a small auxiliary carry-save tree, to minimize hardware.

The performance of the truncated multipliers developed in this thesis have been extensively compared with previously proposed circuits. More than 100 truncated multipliers, with 8 different architectures have been synthesized, with wordlengths ranging from 8 to 32 bits. Area, power and accuracy of the multipliers have been investigated and compared. The analysis shows the proposed LMS1b and LMS2b truncated multipliers represent very often the best trade-off between complexity and accuracy.

Experimental performances, obtained from a test chip in 0.18mm technology have also been presented.

Chapter 4

LMS Truncated Squarer

Squaring function is a fundamental arithmetic operation in many digital signal processing applications such as adaptive filtering, vector quantization, image compression, pattern recognition. A squarer is a multiplier in which the two operands are equal. The resulting circuit can be done simpler than a conventional multiplier since the squarer has actually a single input.

Using the symmetry of Partial Products arising from the equality of two inputs, many techniques have been proposed in order to reduce the area occupation of the squarer, giving however an exact output on $2n$ bits. These techniques will be presented in Sec. 4.1. A further improvement can be obtained in such application which only need a less precise n bits output (see Sec. 4.2), since it is possible to use a truncated squarer, a squarer with n bit input and n bit output. In this case the techniques described in the previous chapter can be extended. In Sec. 4.4 the optimal compensation function, which minimize the mean square error, is presented together with its linear approximation (LMS truncated squarer - Linear Minimum mean Square error truncated squarer). In Sec. 4.5 the results of the simulation and hardware implementation will be presented and compared with the state of the art squarer.

4.1 Folded squarer

The folding technique described in [26] uses the symmetry of the partial product matrix of squarer to achieve 50% reduction of the number of partial products compared with a standard multiplier. The partial products rearrangement technique in [27, 28] can be used to reduce the depth of Partial Products (PPs) Matrix. Ref. [29, 30] joins the Booth technique with folding technique and

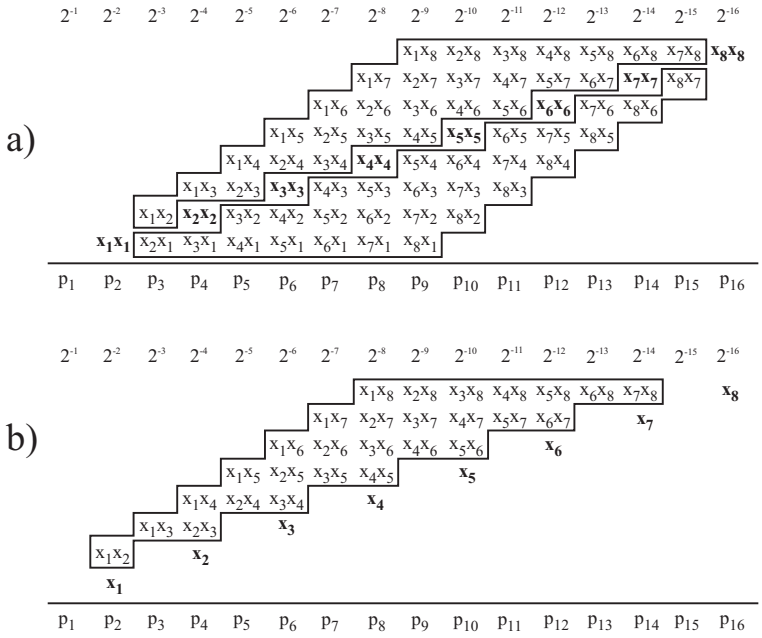

Figure 4.1: Partial Products Matrix of unsigned squarer, n even ($n = 8$). The bold elements form the antidiagonal. a) Full original matrix of the squarer. b) Reduced matrix after applying (4.3).

provides advanced squarer circuits with the design of custom sub-circuits. In this section a description of folding technique will be given.

4.1.1 Unsigned folded squarer

If X is a n-bit unsigned fractional number, and x_i are the bits that represent X we have:

$$X = \sum_{i=1}^{n} x_i \cdot 2^{-i} \tag{4.1}$$

the square of X is:

$$X^2 = \left(\sum_{i=1}^{n} x_i \cdot 2^{-i} \right)^2 = \sum_{i=1}^{n} \sum_{j=1}^{n} x_i x_j \cdot 2^{-i-j} \tag{4.2}$$

Let's consider the case in which n is even. (4.2) is a weighted sum of one bit partial products ($x_i x_j$) that is graphically shown in Fig. 4.1(a). As

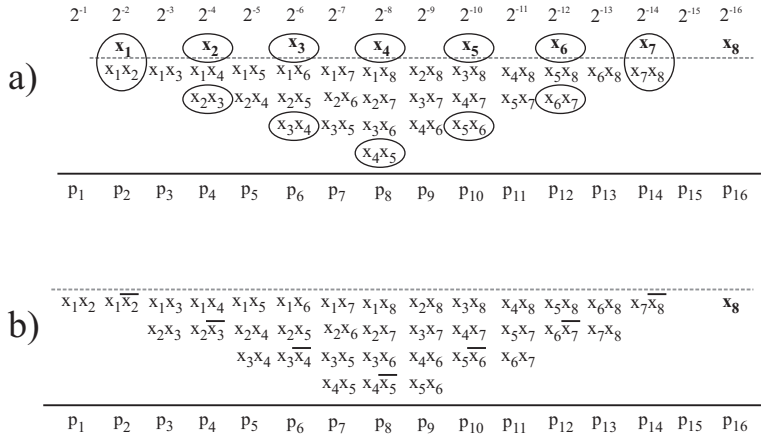

Figure 4.2: Partial Products Matrix of unsigned squarer during the folding process, n even ($n = 8$). a) Partial Product Matrix to which (4.4) can be applied; in each column the circled elements will be grouped. b) Final folded matrix.

$x_i x_j = x_j x_i$, the matrix is symmetric with respect to the antidiagonal [31], shown in Fig. 4.1(a) with boldface character. This means that, with respect to the antidiagonal, the bottom and the top part of can be combined using the identities:

$$\begin{aligned} x_i x_j + x_j x_i &= 2 \cdot x_i x_j \\ x_i x_i &= x_i \end{aligned} \tag{4.3}$$

The result is the PPs matrix shown in Fig. 4.1(b) (n even). Finally the matrix can be further reduced following the technique presented in [32, 27]. Using the identity:

$$x_i + x_i x_{i+1} = 2x_i x_{i+1} + x_i \overline{x_{i+1}} \tag{4.4}$$

in each column the circled elements of the Fig. 4.2(a) can be grouped obtaining the final folded matrix shown in Fig. 4.2(b). The result of the squaring using this modified matrix is given by:

$$X^2 = x_n \cdot 2^{-2n} + \sum_{i=2}^{n} \left(x_i \overline{x_{i-1}} 2^{-2i} + x_i x_{i-1} 2^{-2i-1} \right) + \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} x_i x_j \cdot 2^{-i-j} \tag{4.5}$$

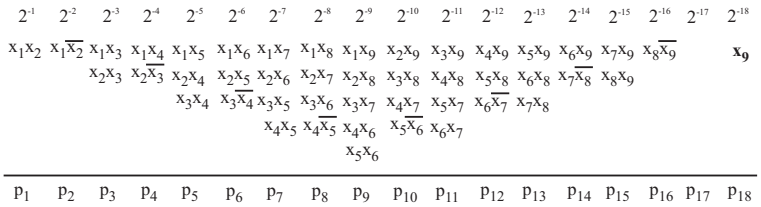

Figure 4.3: Final folded matrix for unsigned squarer, n odd ($n = 9$).

This expression is valid for every n value. Note that, as described in [28], the form of the matrix is slightly different when n is odd (see Fig. 4.3). For every n value, the maximum height of the squarer matrix is $\left\lceil \frac{n}{2} \right\rceil$ and the number of bits in the squaring matrix is $\frac{n^2+n}{2}$. If the squaring is performed using a standard multiplier $X \times X$, the maximum height of the multiplication matrix is n and the number of bits in the matrix in n^2.

4.1.2 Signed folded squarer

If X is a n-bit two's complement fractional number:

$$X = -x_1 \cdot 2^{-1} + \sum_{i=2}^{n} x_i \cdot 2^{-i} \tag{4.6}$$

the square of X is:

$$X^2 = 2^{-1} + 2^{-n} + x_1 \cdot 2^{-2} + \sum_{i=2}^{n}\sum_{j=2}^{n} x_i x_j \cdot 2^{-i-j} + \sum_{i=2}^{n} (\overline{x_i x_1} + \overline{x_1 x_i}) \cdot 2^{-i-1} \tag{4.7}$$

Let's consider the case in which n is even. Following the same reasoning done for the unsigned squarer and applying (4.3) one obtains the matrix of PPs shown in Fig. 4.4(a) (n even). The matrix can be further reduced using the technique presented in [32, 27]. The identity (4.4) can be applied to the black circled elements in Fig. 4.4(a) as done for the unsigned squarer. Furthermore the identity

$$1 + x_i + x_i x_{i+1} = \overline{x_i \overline{x_{i+1}}} + 2x_i \tag{4.8}$$

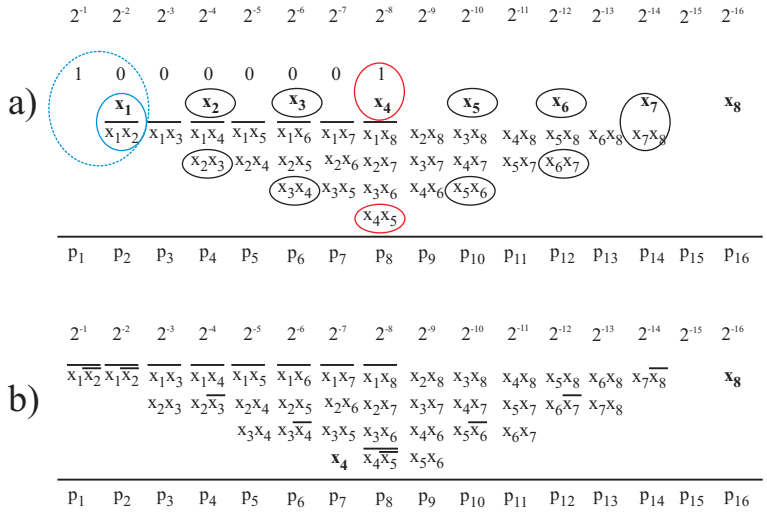

Figure 4.4: Partial Products Matrix of signed squarer during the folding process, n even($n = 8$). a) Partial Product Matrix to which (4.8) can be applied; in each column the circled elements will be grouped. b) Final folded matrix.

is used to modify the bit 1, the terms $a_{\frac{n}{2}}$ and $a_{\frac{n}{2}-1}$ in n^{th} column (red circles in Fig.4.4(a)), and the identities:

$$\begin{aligned} x_i + \overline{x_i x_{i+1}} &= 2x_i\overline{x_{i+1}} + \overline{x_i\overline{x_{i+1}}} \\ 1 + x_i\overline{x_{i+1}} &= 2x_i\overline{x_{i+1}} + \overline{x_i\overline{x_{i+1}}} \end{aligned} \tag{4.9}$$

are used to modify the blue circled elements of the two most significant column obtaining the final folded matrix shown in Fig.4.4(b). If n is odd an identity different from (4.8) is used, and the final matrix is slightly different. This has been described by Wires *et al.* [28].

4.2 Truncated squarer

A $n \times n$ truncated squarer is a squarer in which the output is represented on *n* bits. Since the squarer can be seen as a multiplier with two identical inputs, every consideration for the truncated multiplier is still valid for the truncated squarer. The squarer with *n* bit output, characterized by minimum mean square error, is obtained with a rounding operation. The error is the same calculated

in Sec. 1.3.1; obviously this solution still requires the maximum silicon area, even if the folding technique roughly halves the area occupation required by a squarer with respect to a traditional multiplier. It is however possible to remove some of the partial products in the less significant part of the matrix of Partial Products (PPs) and to introduce a suitable compensation circuit. In this way the approximation error increases but the hardware requirement is reduced. In [33, 34] the authors propose a truncated squarer with variable correction method: the $(n+1)^{th}$ column is added to the n^{th} column and the last $n-1$ columns are discarded. In the following the matrix of partial product before applying the folding identity ((4.4)-(4.8)-(4.9)), that is the reduced matrix before the final step of the folding, will be considered.

The matrix of Partial Product (see Fig. 4.5(a)) can be divided into three subsets: MSP, composed by the first $n-1$ most significant columns, $\text{LSP}_{\text{minor}}$, the last $n-h$ columns, and $\text{LSP}_{\text{major}}$, the columns from $(n+1)^{th}$ to $(n+h)^{th}$. In Ch. 2 it has been demonstrated that $\text{LSP}_{\text{minor}}$ can be estimated with the Input Correction vector ($(n+h+1)^{th}$ column). Due to the presence of the single bit x_i (highlighted in green in Fig. 4.5(a)) the IC will be different if we consider $n_{eq} = n - h$ odd or even. Note that these elements are the only one with the original weight (all the others are doubled), given by 2^{-2i}. Hence they are positioned in the even columns of the matrix of partial product ($2^{nd}, 4^{th}, \ldots$) and they have one bit in common with one of the partial product in the same column and no bit in common with any partial product of the column on the left. For example in Fig.4.5(a) x_5 is present in the column whose weight is 2^{-10} as well as x_5x_6. Instead in the column on the left, whose weight is 2^{-9} no partial product is formed by x_5.

If n_{eq} is even, the $(n+h+1)^{th}$ column (odd) doesn't contain any partial product that is composed of single bit: $x_{\frac{n+h+2}{2}}$ is present in $\text{LSP}_{\text{minor}}$ but not in $(n+h+1)^{th}$ column. Hence, when we try to estimate the contribution to the final result only with $(n+h+1)^{th}$ column, we lose an important information. Thus the IC vector is composed by $(n+h+1)^{th}$ column and the bit $x_{\frac{n+h+2}{2}}$ (see Fig. 4.5(b)).

If n_{eq} is odd, the opposite situation is verified. The $(n+h+1)^{th}$ column (even) contains the bit $x_{\frac{n+h+1}{2}}$, but also a partial product $x_{\frac{n+h+1}{2}}x_{\frac{n+h+1}{2}+1}$ formed by it. Instead of taking twice the same information, $x_{\frac{n+h+1}{2}}$ is one of the element of the IC while $x_{\frac{n+h+1}{2}}x_{\frac{n+h+1}{2}+1}$ is considered a part of $\text{LSP}_{\text{major}}$. In order to not lose the information regarding the other bit of the partial product, also $x_{\frac{n+h+1}{2}+1}$ is considered one of the elements of the IC (see Fig.4.5(c)).

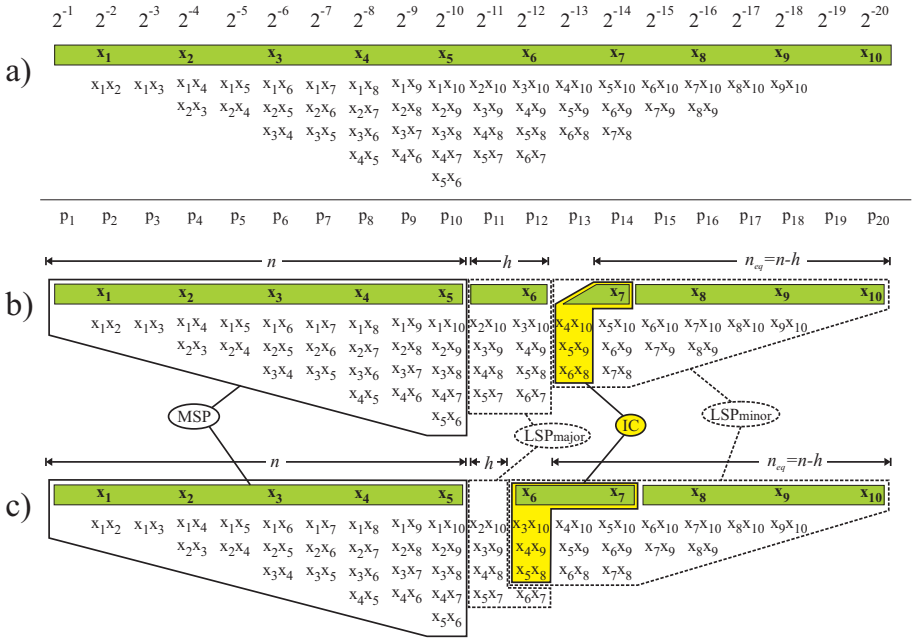

Figure 4.5: Partial Products Matrix of unsigned squarer, ($n = 10$). a) Partial Product Matrix before the last folding operation. In green the matrix elements formed by a single bit. b) Subdivision of the partial product matric in MSP, $\text{LSP}_{\text{major}}$ and $\text{LSP}_{\text{minor}}$ when n_{eq} is even ($n = 10, h = 2$). c) Subdivision of the partial product matric in MSP, $\text{LSP}_{\text{major}}$ and $\text{LSP}_{\text{minor}}$ when n_{eq} is odd ($n = 10, h = 1$).

In the following the unsigned squarer is considered but, as we are referring to the matrix before the final step of folding, all the results are still valid for the signed squarer. In fact the $\text{LSP}_{\text{minor}}$ is the same in both cases since the column from the $n + 1$ to $2n$ are the same for signed and unsigned squarer.

4.3 Optimal compensation function

Considering the subdivision of the matrix, the result can be computed as:

$$X^2 = S_{\text{MSP}} + S_{\text{LSP}_{\text{major}}} + S_{\text{LSP}_{\text{minor}}} \tag{4.10}$$

where S_{MSP}, $S_{\text{LSP}_{\text{major}}}$ and $S_{\text{LSP}_{\text{minor}}}$ are the weighted sum of the elements of the MSP, $\text{LSP}_{\text{major}}$ and $\text{LSP}_{\text{minor}}$ respectively.

In the truncated squarer the terms of the IC are employed to calculate the compensation function $f(\text{IC})$ and the output is computed as:

$$X_t^2 = trunc_n\left(S_{\text{MSP}} + f(\text{IC}) + K_{\text{round}}\right) \tag{4.11}$$

where K_{round} is the rounding constant. Since the same considerations done for the multiplier are still valid for the squarer, for each value of IC the optimal compensation function is $f_{opt}(\text{IC}) = \mu_{\text{LSP}}$, where μ_{LSP} is the mean of $\text{LSP}_{\text{minor}}$ conditioned to the considered value of the IC (see Ch. 2). The optimal compensation function will be computed following the same technique used for the truncated multiplier. Note that even if the reasoning is the same, the optimal $f(\text{IC})$ will be different, not only because the partial products have a different weight, but, most of all, since they have a different probability distribution. Let's see how it will be specialized in the two different cases.

4.3.1 Optimal compensation function $f_{opt}(\text{IC})$ when n_{eq} is even

The elements of the IC (see Fig.4.5(b)) will be indicated as follows:

$$IC = \left[\gamma_1, \gamma_2, \ldots, \gamma_{\frac{n_{eq}}{2}}\right] \tag{4.12}$$

where:

$$\begin{aligned} \gamma_i &= x_{h+1+i} y_{n+1-i} \quad i = 1, \ldots, \frac{n_{eq}}{2} - 1 \\ \gamma_{\frac{n_{eq}}{2}} &= x_{\frac{n+h}{2}+1} \end{aligned} \tag{4.13}$$

In order to calculate the conditioned mean, let's analyze how the elements of $\text{LSP}_{\text{minor}}$ are related to the elements of IC. For instance if we consider Fig. 4.6(a), x_4x_9 depends on $\gamma_2 = x_3x_9$ and $\gamma_3 = x_4x_8$, x_7x_9 depends on $\gamma_2 = x_3x_8$ and $\gamma_4 = x_5x_7$, x_8 depends on $\gamma_3 = x_4x_8$.

In general x_ix_j $(i \neq j)$ depends on two elements of IC, γ_{n+1-j} and γ_t. If $i \leq \dfrac{n+h}{2} + 1$, $\gamma_t = \gamma_{i-h-1}$, otherwise $\gamma_t = \gamma_{n+1-i}$. Note that when $i = \frac{n+h}{2} + 1$ the partial product is correlated to the element of the IC with the single bit.

The generic x_j depends only on γ_{n+1-j}.

In order to simplify the analytical calculation, let's indicate each element of $\text{LSP}_{\text{minor}}$ with the index of the elements of the IC to whom they are correlated: instead of x_ix_j, let's use $p_{t,n+1-j}$ (see Fig. 4.6(b)).

$\text{LSP}_{\text{minor}}$ can be divided in 4 parts (see Fig. 4.6(b)):

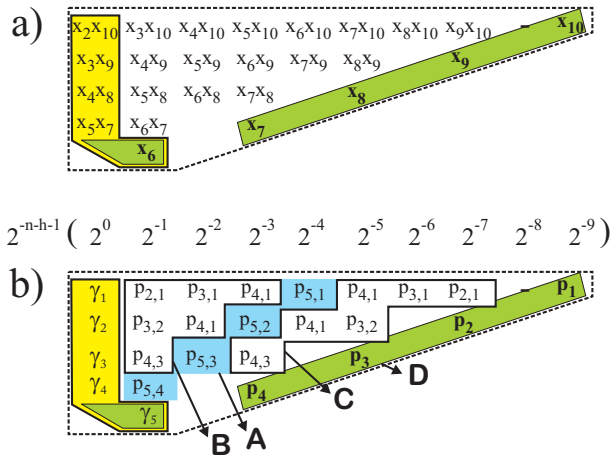

Figure 4.6: $\text{LSP}_{\text{minor}}$ of unsigned squarer, ($n = 10$, $h = 0$). a) $\text{LSP}_{\text{minor}}$. In yellow, elements of IC vector, in green elements of the matrix composed by a single bit. b) $\text{LSP}_{\text{minor}}$ in which the dependence by the elements of IC is shown.

A the diagonal correlated to the only one element of IC composed by a single bit ($\gamma_{\frac{n_{eq}}{2}}$);

B the partial products on the left side respect to A;

C the partial products on the right side respect to A;

D the partial products composed by only one bit.

Following this partition, the sum of the elements of $\text{LSP}_{\text{minor}}$ can be written as:

$$S_{\text{LSP}_{\text{minor}}} = S_{\text{IC}} + S_A + S_B + S_C + S_D \tag{4.14}$$

where:

$$S_{\text{IC}} = 2^{-n-h-1} \sum_{i=1}^{\frac{n_{eq}}{2}} \gamma_i \tag{4.15}$$

$$S_A = 2^{-n-h-1} \sum_{i=1}^{\frac{n_{eq}}{2}-1} p_{\text{A}\,\frac{n_{eq}}{2},i} \cdot 2^{-\frac{n_{eq}}{2}+i} \tag{4.16}$$

$$S_B = 2^{-n-h-1} \sum_{i=2}^{\frac{n_{eq}}{2}-1} \sum_{j=1}^{i-1} p\mathrm{B}_{i,j} \cdot 2^{-i+j} \tag{4.17}$$

$$S_C = 2^{-n-h-1} \sum_{i=2}^{\frac{n_{eq}}{2}-1} \sum_{j=1}^{i-1} p\mathrm{C}_{i,j} \cdot 2^{-n_{eq}+i+j} \tag{4.18}$$

$$S_D = 2^{-n-h-1} \sum_{i=1}^{\frac{n_{eq}}{2}-1} p\mathrm{D}_i \cdot 2^{-n_{eq}+2i-1} \tag{4.19}$$

The last step is to calculate the mean of the generic element of $\mathrm{LSP}_{\mathrm{minor}}$. In the hypotheses of input bits independent and identically distributed, as $x_i y_j$ is correlated to different elements of the IC its mean value is given by:

$$\underset{IC=A}{E}\{x_i y_j\} = \underset{IC=A}{E}\{x_i\} \cdot \underset{IC=A}{E}\{y_j\} \tag{4.20}$$

If we consider a generic element $p_{i,j}$ of B or C, it is correlated to two different elements of IC given by an *AND* operation between its bits. Hence if $\gamma_j = 1$, x_{n+1-j} will be 1 with probability equal to 1, otherwise with a probability equal to $1/3$ (one of the remaining cases):

$$\underset{IC=A}{E}\{x_{n+1-j}\} = \frac{1}{3}(1 + 2\gamma_j) \tag{4.21}$$

Analogous reasoning is valid for the other element of partial product. Hence the mean value of a generic element of $\mathrm{LSP}_{\mathrm{minor}}$ can be expressed as ($X = B, C$):

$$\underset{IC=A}{E}\{p\mathrm{X}_{i,j}\} = \frac{1}{9}(1 + 2\gamma_i)(1 + 2\gamma_j) \in \left\{\frac{1}{9}, \frac{1}{3}, 1\right\} \tag{4.22}$$

When the elements of A are considered, the mean value changes. In fact one of the bit of each partial product is correlated to the single bit element of IC, $\gamma_{\frac{n-h}{2}}$; hence for each fixed value of IC its value is a constant:

$$\underset{IC=A}{E}\{p\mathrm{A}_{i,j}\} = \gamma_i \frac{1}{3}(1 + 2\gamma_j) \in \left\{\frac{1}{3}\gamma_i, \gamma_i\right\} \tag{4.23}$$

Finally let's consider the elements of D. These elements are composed by a single bit and are correlated to a single element of IC, then:

$$\underset{IC=A}{E}\{p\mathrm{D}_i\} = \frac{1}{3}(1 + 2\gamma_i) \in \left\{\frac{1}{3}, 1\right\} \tag{4.24}$$

Using (4.22)-(4.23)-(4.24) in (4.16)-(4.17)-(4.18)-(4.19) and then the obtained value in eq.(4.14), with some algebra one obtains:

$$f_{opt}(\text{IC}) = 2^{-n-h-1}\left(K + \sum_{i=1}^{\frac{n_{eq}}{2}} f_i\gamma_i + \sum_{i=1}^{\frac{n_{eq}}{2}-1}\sum_{j=i+1}^{\frac{n_{eq}}{2}} f_{ij}\gamma_i\gamma_j\right) \tag{4.25}$$

where:

$$\begin{aligned}
K &= \frac{1}{9}\left(\frac{n_{eq}}{2} + 2^{1-\frac{n_{eq}}{2}} + \frac{1}{3}2^{1-n_{eq}} - \frac{13}{6}\right)\\
f_1 &= \frac{1}{9}\left(11 - 2^{2-n_{eq}} - 2^{2-\frac{n_{eq}}{2}}\right)\\
f_i &= \frac{1}{9}\left(13 - 2^{2-i} - 2^{i-n_{eq}}(2^{1+\frac{n_{eq}}{2}} + 4 - 2^i)\right) \quad i = 2\ldots\frac{n_{eq}}{2} - 2\\
f_{\frac{n_{eq}}{2}-1} &= \frac{1}{9}\left(\frac{49}{4} - 5\cdot 2^{1-\frac{n_{eq}}{2}}\right)\\
f_{\frac{n_{eq}}{2}} &= \frac{1}{3}\left(\frac{5}{2} - 2^{1-\frac{n_{eq}}{2}}\right)\\
f_{ij} &= \frac{4}{9}\left(2^{i-j} + 2^{-n_{eq}+i+j}\right) \quad i,j = 1\ldots\frac{n_{eq}}{2} - 1\\
f_{i,\frac{n_{eq}}{2}} &= \frac{1}{3}\left(2^{1+i-\frac{n_{eq}}{2}}\right) \quad i,j = 1\ldots\frac{n_{eq}}{2} - 1
\end{aligned} \tag{4.26}$$

4.3.2 Optimal compensation function $f_{opt}(\text{IC})$ when n_{eq} is odd

The elements of the IC (see Fig.4.5(c)) will be indicated as follows:

$$IC = [\gamma_1, \gamma_2, \ldots, \gamma_w] \tag{4.27}$$

where:

$$\begin{aligned}
w &= \frac{n_{eq}+1}{2}\\
\gamma_i &= x_{h+1+i}y_{n+1-i} \quad i = 1,\ldots,w-2\\
\gamma_{w-1} &= x_{w+h}\\
\gamma_w &= x_{w+h+1}
\end{aligned} \tag{4.28}$$

As done previously, let's indicate each element of $\text{LSP}_{\text{minor}}$ with the index of the elements of the IC to whom they are correlated: instead of x_ix_j, let's use $p_{t,n+1-j}$ (see Fig.4.7(b)). $\text{LSP}_{\text{minor}}$ can be divided in 5 parts:

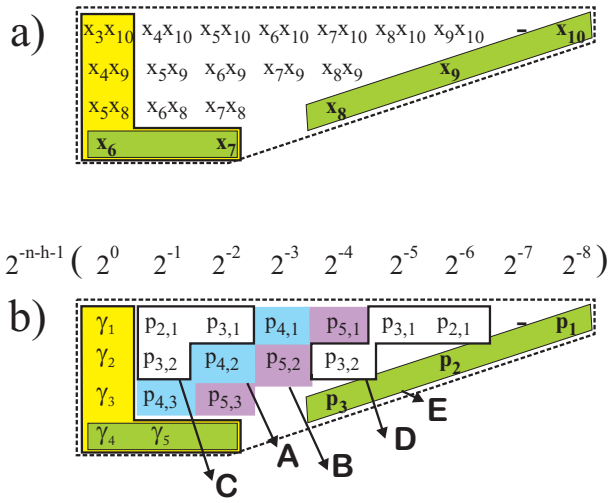

Figure 4.7: $\text{LSP}_{\text{minor}}$ of unsigned squarer, ($n = 10$, $h = 1$). a) $\text{LSP}_{\text{minor}}$. In yellow, elements of IC vector, in green elements of the matrix formed by a single bit. b) $\text{LSP}_{\text{minor}}$ in which the dependence by the elements of IC is shown.

A the diagonal correlated to γ_{w-1};

B the diagonal correlated to γ_w;

C the partial products on the left side respect to A;

D the partial products on the right side respect to B;

E the partial products composed by only one bit.

Following this partition, the sum of the elements of $\text{LSP}_{\text{minor}}$ can be written as:

$$S_{\text{LSP}_{\text{minor}}} = S_{\text{IC}} + S_A + S_B + S_C + S_D + S_E \tag{4.29}$$

where:

$$S_{\text{IC}} = 2^{-n-h-1} \sum_{i=1}^{w} \gamma_i \tag{4.30}$$

$$S_A = 2^{-n-h-1} \sum_{i=1}^{w-2} p\text{A}_{w-1,i} \cdot 2^{-w+i+1} \tag{4.31}$$

$$S_B = 2^{-n-h-1} \sum_{i=1}^{w-2} p\mathrm{B}_{w,i} \cdot 2^{-w+i} \tag{4.32}$$

$$S_C = 2^{-n-h-1} \sum_{i=2}^{w-2} \sum_{j=1}^{i-1} p\mathrm{C}_{i,j} \cdot 2^{-i+j} \tag{4.33}$$

$$S_D = 2^{-n-h-1} \sum_{i=2}^{w-2} \sum_{j=1}^{i-1} p\mathrm{D}_{i,j} \cdot 2^{-2w+i+j+1} \tag{4.34}$$

$$S_E = 2^{-n-h-1} \sum_{i=1}^{w-2} p\mathrm{E}_i \cdot 2^{-2w+2i} \tag{4.35}$$

The mean value of each elements of $\mathrm{LSP}_{\mathrm{minor}}$ can be calculated in the same way done previously, obtaining:

$$\underset{IC=A}{E} \{p[\text{C-D}]_{i,j}\} = \frac{1}{9} (1 + 2\gamma_i) (1 + 2\gamma_j) \in \left\{ \frac{1}{9}, \frac{1}{3}, 1 \right\} \tag{4.36}$$

$$\underset{IC=A}{E} \{p[\text{A-B}]_{i,j}\} = \frac{1}{3} (1 + 2\gamma_j) \gamma_i \in \left\{ \frac{1}{3}\gamma_i, \gamma_i \right\} \tag{4.37}$$

$$\underset{IC=A}{E} \{p\mathrm{E}_i\} = \frac{1}{3} (1 + 2\gamma_i) \in \left\{ \frac{1}{3}, 1 \right\} \tag{4.38}$$

Using (4.36)-(4.37)-(4.38) in (4.31)-(4.32)-(4.33)-(4.34)-(4.35) and then the obtained value in eq.(4.29), with some algebra one obtains:

$$f_{opt}(\mathrm{IC}) = 2^{-n-h-1} \left(K + \sum_{i=1}^{w} f_i \gamma_i + \sum_{i=1}^{w-1} \sum_{j=i+1}^{w} f_{ij} \gamma_i \gamma_j \right) \tag{4.39}$$

where:

$$\begin{aligned}
K &= \frac{1}{9}\left(\frac{1}{12}\left(-43+92^{3-w}+2^{4-2w}\right)+w\right)\\
f_1 &= \frac{1}{9}\left(11-8*2^{-2w}-32^{2-w}\right)\\
f_i &= \frac{1}{9}\left(-2^{4-2w}-2^{1+2i-2w}+2^{3-w}-32^{1+i-w}+13-2^{2-i}\right)\\
f_{w-2} &= \frac{11}{8}-2^{1-w}\\
f_{w-1} &= \frac{4}{3}\left(1-2^{-w}\right)\\
f_w &= \frac{5}{12}-\frac{1}{3}2^{1-w}\\
f_{ij} &= \frac{4}{9}\left(2^{i-j}+2^{-2w+i+j+1}\right) \quad i,j=1\ldots w-2\\
f_{i,w-1} &= \frac{1}{3}\left(2^{1+i-w}\right) \quad i=1\ldots w-2\\
f_{i,w} &= \frac{1}{3}\left(2^{i-w}\right) \quad i=1\ldots w-2
\end{aligned} \tag{4.40}$$

4.4 Optimal Linear Compensation function

The optimal compensation function can be hardly implemented in hardware, since it is a quadratic form of the elements of the IC. In Ch. 2 it has been demonstrated that the only linear combination of the elements of the IC is a good approximation of the optimum function. Hence let's consider

$$f_{lin}(\mathrm{IC}) = 2^{-n-h-1}\left(K_l+\sum_{i=2}^{z} l_i\gamma_i\right) \tag{4.41}$$

where $z=\frac{n-h}{2}$ if n_{eq} is even, otherwise to $z=w=\frac{n_{eq}+1}{2}$. In order to calculate the optimal coefficients and the optimal constant for $f_{lin}(\mathrm{IC})$ it is possible to conduct a demonstration similar to the one presented for the truncated multiplier. Here only the fundamental steps will be recalled, and the differences will be highlighted.

The optimal values of the coefficients K_l, l_i in (4.41) are obtained by minimizing σ^2_{lin} (2.38), given by:

$$\sigma^2_{\mathrm{lin}} = \sum_{A\in\Theta}\left(f_{lin}(A)-\mu_{\mathrm{LSP}}(A)-\mu_{\mathrm{lin}}\right)^2 P(A) \tag{4.42}$$

where:

$$\mu_{\text{lin}} = \sum_{A \in \Theta} \left(f_{lin}(A) - \mu_{\text{LSP}}(A) \right) \cdot P(A) \tag{4.43}$$

The problem can be solved as shown in Ch. 2 (see eq. 2.62), with one important difference. When we deal with the truncated multiplier all the elements of the IC vector are partial product obtained with an *AND* operation between the bits, hence their probability of being 1 is 1/4. In the truncated squarer some of elements of the IC are constituted by only one bit, in this case the probability of being 1 is 1/2.

Solving the linear system equivalent to (2.62) the coefficients of the linear function are computed in closed form and imposing $\mu_{\text{lin}} = 0$ also the expression of the constant will be obtained in closed form. The expression are the following.

n_{eq} *even*

$$\begin{aligned}
K_l &= \frac{1}{27}\left(3\frac{n_{eq}}{2} + 212^{-1-\frac{n_{eq}}{2}} + 2^{1-n_{eq}} - \frac{35}{4}\right) \\
l_1 &= \frac{4}{3}\left(1 - 2^{-n_{eq}}\right) \\
l_i &= \frac{1}{3}\left(5 - 2^{1-i} - 2^{1+i-n_{eq}}\right) \quad i = 2 \ldots \frac{n_{eq}}{2} - 2 \\
l_{\frac{n_{eq}}{2}-1} &= \frac{5}{3}\left(1 - 2^{-\frac{n_{eq}}{2}}\right) \\
l_{\frac{n_{eq}}{2}} &= 1 - 2^{-\frac{n_{eq}}{2}}
\end{aligned} \tag{4.44}$$

n_{eq} *odd*

$$\begin{aligned}
K_l &= \frac{1}{48}\left(-17 + 9 \cdot 2^{2-w} + 4w\right) \\
l_1 &= \frac{1}{9}\left(12 - 24 \cdot 2^{-2w} - 9 \cdot 2^{-w}\right) \\
l_i &= \frac{1}{9}\left(-2^{-2w}\left(2^4 + 2^{2+i} + 2^{2+2i}\right) + 2^{-w}\left(2^3 - 9 \cdot 2^{-1+i}\right) + 15 - 3 \cdot 2^{1-i}\right) \\
l_{w-2} &= \frac{7}{4} - 3 \cdot 2^{-w} \\
l_{w-1} &= \frac{3}{2} - 2^{1-w} \\
l_w &= \frac{1}{2} - 2^{-w}
\end{aligned} \tag{4.45}$$

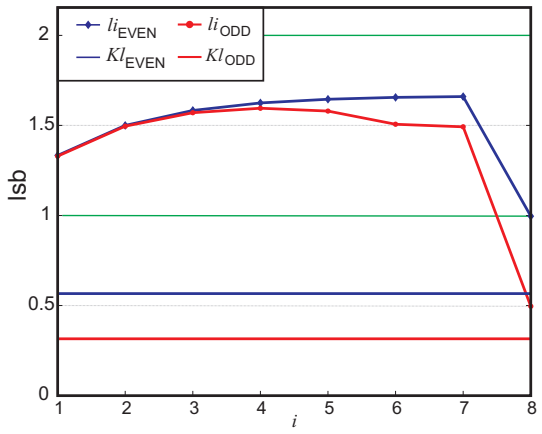

Figure 4.8: Optimal coefficients and constant of $f_{lin}(\mathrm{IC})$ when n_{eq} is odd (red line) and even (blue line). The green lines represents the possible quantization levels.

The last step is the quantization of the coefficient. In Ch. 2 it has been demonstrated that it is a problem of discrete optimization, not simple to solve. Here only the result of the quantization on 1 bit are reported (see Fig. 4.8):

n_{eq} even

$$\begin{aligned} K_{\text{round}} &= \mathbf{lsb_{IC}} \\ q_i &= 1 \quad i = 1, 2, \frac{n_{eq}}{2} \\ q_i &= 2 \quad i = 3, \ldots, \frac{n_{eq}}{2} - 1 \end{aligned} \tag{4.46}$$

n_{eq} odd

$$\begin{aligned} K_{\text{round}} &= \mathbf{lsb_{IC}} \\ q_i &= 1 \quad i = 1, 2, \frac{n_{eq}+1}{2} - 1 \\ q_i &= 2 \quad i = 3, \ldots, \frac{n_{eq}+1}{2} - 2 \\ q_{\frac{n_{eq}+1}{2}} &= 0 \end{aligned} \tag{4.47}$$

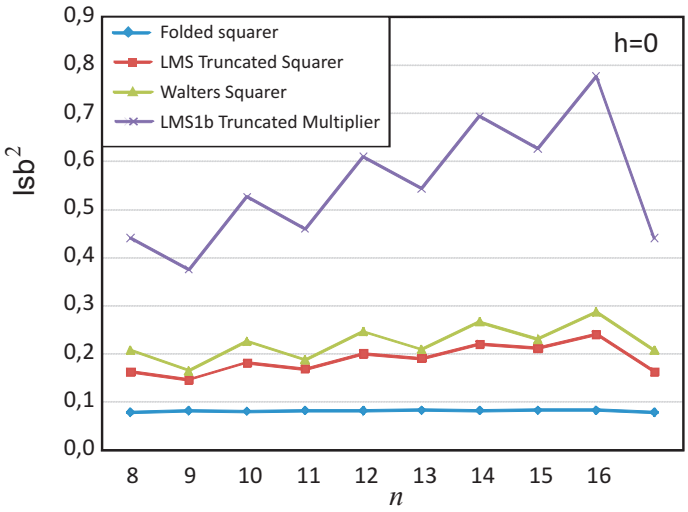

Figure 4.9: Mean square error obtained using a full-width folded squarer with final rounding (blue line§), LMS truncated squarer (red line), Walters [33] squarer (green line) and LMS1b truncated multiplier (violet line), varying n for $h = 0$.

Given these coefficients the implementation results to be very simple. As done for LMS1b truncated multiplier, the proposed circuit eliminates the $\text{LSP}_{\text{minor}}$ matrix and simply reorganizes the IC vector according to (4.46) or (4.47).

4.5 VLSI Implementation

To verify the performance of the proposed circuits, various squares have been implemented in TSMC 0.18μm technology. As the error strongly depends on the bit width of the multiplier (there are also some differences if n_{eq} is odd or even), for the implementation n values ranging from 8 to 16 have been considered with $h = 0, 1$.

Figs. 4.9-4.10 show the mean square error obtained by the full-width folded squarer (output rounded to n bits), the proposed LMS squarer, the squarer proposed by Walters et al. [33]. For comparison we have also analyzed the results given by LMS1b truncated multipliers, used as a squarer.

The data reported in the figures show that using the LMS1b truncated mul-

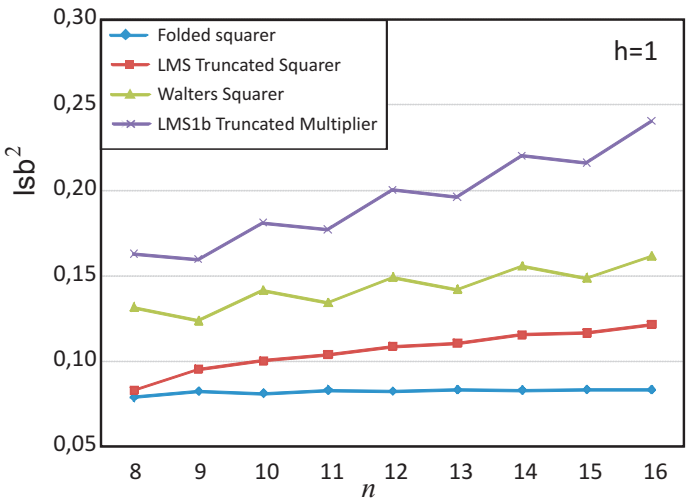

Figure 4.10: Mean square error obtained using a full-width folded squarer with final rounding (blue line§), LMS truncated squarer (red line), Walters [33] squarer (green line) and LMS1b truncated multiplier (violet line), varying n for $h = 1$.

tiplier as a squarer does not provide any advantages with respect to the folding technique [27]. Better results are provided with truncated squares as the one of [33] and the proposed squarer. In particular the LMS truncated squarer gives the best approximation and also slightly improved in power and maximum frequency. Note that the result is different if we consider n_{eq} odd or even. If n_{eq} is even the proposed technique gives an error 20% (on average) lower than [33]. If we consider the odd case this error is only $8-10\%$ (on average) lower. This difference can be explained if we consider the IC vector. In [33] Walters *et al* uses $\lceil \frac{n_{eq}}{2} \rceil$ PPs in order to estimate the LSP part. LMS truncated squarer always consider $\lfloor \frac{n_{eq}}{2} \rfloor$ PPs. So, if n_{eq} is odd it estimates the LSP with a lower number of PPs. For any considered value of n and h the proposed truncated squarer always gives the best result.

Finally the Tab. 4.1 shows the mean error, the mean square error ($\mathbf{lsb} = 2^{-n}$), the maximum working frequency, the silicon area occupation and the power dissipation. Compared with other truncated squarer circuits proposed in the literature, LMS truncated squarer reduces the mean square error, the power dissipation, the silicon area occupation, and increases the maximum working frequency.

Table 4.1: Comparison between squarer, $h = 0$

n	Architecture	$\overline{\varepsilon}$ lsb	ε_{ms} lsb^2	Max Freq MHz	Area $10^3 \mu m^2$	Power $\frac{\mu W}{MHz}$
9	Folded [27]	0.028	0.082	481	3157	7.3
	LMS1b truncated multipliers	0.167	0.375	478	3100	9.5
	Walters [33]	0.083	0.164	515	2112	5.1
	LMS squarer	**0.072**	**0.145**	**524**	**1986**	**4.9**
10	Folded [27]	0.014	0.082	476	3875	9.3
	LMS1b truncated multipliers	0.333	0.529	439	3829	12.1
	Walters [33]	0.166	0.226	498	2565	6.5
	LMS squarer	**0.166**	**0.181**	**500**	**2425**	**6.3**
11	Folded [27]	0.015	0.082	435	4797	11.7
	LMS1b truncated multipliers	0.167	0.460	397	4627	15.1
	Walters [33]	0.083	0.187	495	3057	7.9
	LMS squarer	**0.073**	**0.169**	**493**	**2877**	**7.8**
12	Folded [27]	0.007	0.082	418	5715	14.3
	LMS1b truncated multipliers	0.333	0.607	398	5495	18.3
	Walters [33]	0.167	0.246	469	3579	9.6
	LMS squarer	**0.167**	**0.200**	**476**	**3380**	**9.5**
13	Folded [27]	0.007	0.082	403	6749	17.4
	LMS1b truncated multipliers	0.168	0.543	389	6427	21.7
	Walters [33]	0.083	0.209	437	4135	11.4
	LMS squarer	**0.072**	**0.190**	**433**	**3895**	**11.1**
14	Folded [27]	0.004	0.082	398	7780	20.5
	LMS1b truncated multipliers	0.333	0.693	376	7428	25.5
	Walters [33]	0.167	0.266	422	4720	13.3
	LMS squarer	**0.167**	**0.220**	**427**	**4467**	**13.2**
15	Folded [27]	0.004	0.082	385	8995	24.2
	LMS1b truncated multipliers	0.167	0.626	366	8499	29.4
	Walters [33]	0.083	0.230	420	5346	15.4
	LMS squarer	**0.072**	**0.212**	**422**	**5053**	**15.1**
16	Folded [27]	0.002	0.082	365	10205	27.9
	LMS1b truncated multipliers	0.333	0.776	356	9640	33.8
	Walters [33]	0.167	0.287	413	6001	17.6
	LMS squarer	**0.167**	**0.240**	**417**	**5731**	**17.5**

Chapter 5

FIR filter

Digital filters, that shape the frequency spectrum of the signal, are an important part of DSP and are used for audio, speech, video, image processing, etc.

Digital filters can be classified in FIR filter and IIR filter. Indicating with $x[n]$ and $y[n]$ the input and the output of the filter at the n^{th} instant, the FIR filter computes the output as a function of previous inputs ($y_{FIR}[n] = f(x[0], \ldots, x[n])$), while in IIR filter as a function of previous inputs and previous outputs ($y_{IIR}[n] = f(x[0], \ldots, x[n], y[1], \ldots, y[n-1])$). In this chapter the attention is focused on FIR filters because, even if sometimes they require more memory and calculations to achieve a given frequency response, they present some advantages. FIR filters can easily be designed to be "linear phase" (they delay the input signal but do not distort its phase); they are suited to multi rate application, as decimation and interpolation, since in the FIR filters some calculations can be omitted while IIR filters require that each output, even if discarded, must be individually calculated (due to the feedback). Finally IIR filters can cause significant problems due to the use of feedback and hence a small error influences every subsequent output.

The basic operation of a digital filter is MAC (Multiply and ACcumulate). MAC is the operation of multiplying a coefficient by the corresponding delayed data sample and accumulating the result, hence the MAC operation needs a multiplier and an accumulator. Since the multiplication is the basic operation, in this chapter the impact of the use of truncated multiplier in conventional filtering applications (such as a pass-band, a low-pass and a high-pass filter) is investigated. Aim of the analysis is the definition of the parameters of the truncated multipliers that are more significant for the optimization of a digital

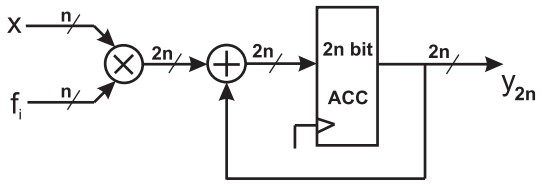

Figure 5.1: MAC with n bits inputs and $2n$ bits output. The error provided by this configuration is zero.

filter (Sec. 5.1), the evaluation of the performances of the proposed LMS multiplier when applied to a digital filter and the comparison of the performances in terms of error, power, frequency and area occupation with the use of existent truncated multiplier (Sec. 5.2).

5.1 FIR filter with truncated MAC

The output of a FIR filter is computed as:

$$y[n] = \sum_{i=0}^{N-1} h_i \cdot x[n-i] \tag{5.1}$$

The FIR filter multiplies an array of the most recent N data samples by an array of constants (the tap coefficients, h_i), and sums the resulting elements. The number of FIR taps is related to the amount of calculations and memory required to implement the filter, and to the amount of "filtering" that the FIR provides. More taps means increased stopband attenuation, reduced ripple, narrower filters, etc.

A description of FIR filter design techniques is beyond the scope of this dissertation (many good references are available [35, 36, 37]); in this chapter the attention is focused on the multiplier since the multiplication is the dominant computational requirement. The basic MAC operation is shown in (Fig. 5.1). It is composed by a multiplier $n \times n$ and an accumulator on $2n$ bits. The final output is represented on $2n$ bits.

Due to the presence of addition operation, overflow can occur. The methodology used to deal with overflow should depend on the specifics of the signal, the type of operation and the DSP architecture used. In general, overflow handling methodologies can be classified in five categories: saturation, input scaling, fixed scaling, dynamic scaling and system design consider-

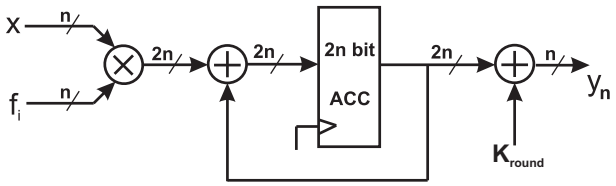

Figure 5.2: MAC with n bits inputs and n bits output. It uses a full-width multiplier and the output is rounded to n bits summing a rounding constant to the $2n$ bits output. In this configuration a rounding error is present.

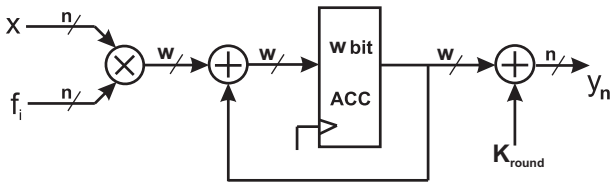

Figure 5.3: MAC with n bits inputs, w bits output. It uses a truncated multiplier with n bits inputs and $w < 2n$ bit output. This configuration presents the rounding error as in Fig. 5.2 and the error introduced by the truncated multiplier.

ations. In this chapter the scaling solution will be used, in order to consider a solution which is independent of the architecture and avoid the introduction of additional hardware which could hide the differences between the implementations with various truncated multipliers.

When a MAC operation with n bits precision is required, the lowest error is obtained working with a full precision until the final step, when the output of the accumulator is reported on n bit with a rounding operation (see Fig. 5.2). In this configuration only a rounding error is present.

Another possibility is to work with a lower precision in the MAC unit itself. The full width multiplier can be replaced by a truncated multiplier whose output is left on $w = n + h$ bits (see Fig. 5.3). This solution reduces the area occupation and power dissipation of the MAC unit, but introduces a final error higher than previous one, since the rounding error is added to the error introduced by each multiplication. Indicating with:
y_{2n} the output of configuration in Fig. 5.1, full precision MAC,

y_n the output of configuration in Fig. 5.2-5.3, truncated MAC,
the output of FIR filter implemented with a truncated MAC is affected by an error:

$$\varepsilon_{FIR} = y_{2n} - y_n \tag{5.2}$$

As error metric it will be considered the mean square error $\varepsilon^2_{\text{FIR}}$ and the mean error μ_{FIR}, normalized to the weight of the less significant bit of the output filter ($\mathbf{lsb} = 2^{-n}$):

$$\varepsilon^2_{\text{FIR}} = \frac{E[(e_{FIR})^2]}{\mathbf{lsb}^2} = \frac{E[(e_{FIR})^2]}{2^{-2n}} \tag{5.3}$$

$$\mu_{\text{FIR}} = \frac{E[e_{FIR}]}{\mathbf{lsb}} = \frac{E[e_{FIR}]}{2^{-n}} \tag{5.4}$$

Using the configuration of Fig. 5.2, only a rounding error is present: $e_{FIR} = e_{Round}$. The mean and mean square error of the e_{Round} has been computed in Sec. 2.2.1, hence the error of the FIR filter has the following statistical characterization:

$$\mu_{\text{FIR}} = \mu_{\text{round}} = 0 \tag{5.5}$$

$$\varepsilon^2_{\text{FIR}} = \varepsilon^2_{\text{round}} = \frac{1}{12}\,\mathbf{lsb}^2 = \frac{1}{12}\cdot 2^{-2n} \tag{5.6}$$

These are the lower bound for any FIR filter with an output formed by the same number of bits of the input and will be considered as the reference circuit in the following.

Using a truncated multiplier, each operation of multiplication introduces an error e_{Mi}, which depends by the coefficient f_i and the applied input. Finally a further error, e_{round}, is due to the final rounding (in order to get *n*-bit output from *w*-bit output). The error of the filter is therefore equal to:

$$e_{FIR} = \sum_{i=1}^{N} e_{Mi} + e_{\text{round}} \tag{5.7}$$

Assuming the independence of the e_{Mi} and of the e_{round}:

$$\mu_{\text{FIR}} = E\left[\sum_{i=1}^{N} e_{Mi} + e_{\text{round}}\right] = \sum_{i=1}^{N} E[e_{Mi}] + E[e_{\text{round}}] =$$
$$= \sum_{i=1}^{N} E[e_M|f_i] + \mu_{\text{round}} \tag{5.8}$$

$$\sigma^2_{\text{FIR}} = \text{VAR}\left[\sum_{i=1}^{N} e_{Mi} + e_{\text{round}}\right] = \sum_{i=1}^{N} \text{VAR}[e_{Mi}] + \text{VAR}[e_{\text{round}}] =$$
$$= \sum_{i=1}^{N} \text{VAR}[e_M|f_i] + \sigma^2_{\text{round}} \tag{5.9}$$

where $E[e_M|f_i]$ and $\text{VAR}[e_M|f_i]$ are respectively the mean error and variance of the error of the truncated multiplier given an input equal to f_i. Finally, given that $\sigma^2_{\text{round}} = \varepsilon^2_{\text{FIR}} = \frac{1}{12}$ (since $\mu_{\text{round}} = 0$), using (5.5)-(5.6) in (5.8)-(5.9), the statistical characterization of the FIR filter is:

$$\mu_{\text{FIR}} = \sum_{i=1}^{N} E[e_M|f_i] \tag{5.10}$$

$$\varepsilon^2_{\text{FIR}} = \sigma^2_{FIR} + (\mu_{\text{FIR}})^2 = \frac{1}{12} + \sum_{i=1}^{N} \text{VAR}[e_M|f_i] + \left(\sum_{i=1}^{N} E[e_M|f_i]\right)^2 \tag{5.11}$$

Note that the (5.10)-(5.11) represent the theoretical FIR error applying all possible inputs to the FIR filter, that is, applying a random variable with an uniform distribution. Otherwise the probability distribution function (pdf) of the signal can influence the result. In particular, the error of the filter depends on the mean error and the variance of the error of the truncated multiplier conditioned to the input with the particular probability distribution. In this case the error is equal to:

$$\mu_{\text{FIR}} = \sum_{i=1}^{N} E[e_M|f_i, x] \tag{5.12}$$

$$\varepsilon^2_{\text{FIR}} = \sigma^2_{\text{FIR}} + (\mu_{\text{FIR}})^2 =$$
$$= \frac{1}{12} + \sum_{i=1}^{N} \text{VAR}[e_M|f_i, x] + \left(\sum_{i=1}^{N} E[e_M|f_i, x]\right)^2 \tag{5.13}$$

(5.10)-(5.11) show that a truncated multiplier can replace a full-width multiplier (saving area and power) if

$$\sum_{i=1}^{N} \text{VAR}[e_M|f_i] + \left(\sum_{i=1}^{N} E[e_M|f_i]\right)^2 \ll \frac{1}{12} = \varepsilon^2_{FIR2n} \tag{5.14}$$

Table 5.1: Characteristic of the considered FIR Filters.

	LOW-PASS	BAND-PASS	HIGH-PASS
Sampling Frequency	48KHz	48KHz	48KHz
Passband Frequency	1KHZ	8.5KHz-12KHz	10KHz
Stopband Frequency	2KHz	6KHz-14.5KHz	9.5KHz
Passband ripple	1dB	1dB	1dB
Stopband attenuation	60dB	80dB - 80dB	80dB
Filter order (N)	100	49	265

where ε^2_{FIR2n} is the error committed in configuration of Fig. 5.2. In fact, if the condition of (5.14) is verified, the error introduced by the truncated multiplier is negligible respect to the error introduced by the final rounding.

The first point that comes out from the equations (5.10)-(5.11) and from the more general equations (5.12)-(5.13) is that in this kind of application it is very important to have truncated multiplier characterized by a low variance but also a low mean error.

Furthermore the condition (5.14) is very useful in the design of the optimal compensation function of a truncated multiplier employed in the MAC unit. In fact in Ch. 2 the optimal compensation function has been calculated in the hypothesis of bits independent and identically distributed. Using a different pdf (which depends on the value that the coefficients h_i of the filter can assume), the mean of the discarded matrix will change according to the new probability distribution of the bits, hence a new optimal compensation function can be computed.

5.2 VLSI implementation

5.2.1 FIR synthesis

In order to analyze the impact of the use of fixed-with multiplier for the realization of FIR filters, three filters, implemented with a different number N of taps (see Tab. 5.1), have been considered.

As the error strongly depends on the bit width of the multiplier, different n values have been considered for the implementation of the filter: $n = 12, 16, 20$. The input is assumed to be a signed fractional number rep-

resented in 2's complement, with the weights of the MSB and the LSB respectively equal to 2^0 and $2^{-(n-1)}$ (indicated as $Q_{0.(n-1)}$); filter coefficients have been quantized on *n* bits in $Q_{0.(n-1)}$ representation. The output is also a signed fractional number represented on $n+h$ bits ($Q_{1.(n-2+h)}$). For $n = 16$, each of the low-pass, band-pass and high-pass filters has been implemented in TSMC 0.18μm technology using a full-width multiplier (output rounded to $w = n+h$ bits), the Jou *et al.* [13] multiplier [JOU], the Curticapean *et al.* [14] multiplier [CCP], the Van *et al.* [15, 16] multiplier [VAN] and the proposed LMS1b multiplier. This allows the comparison of the performances between the various multipliers in terms of error, area occupation, maximum working frequency and power dissipation. For $n = 12$ and $n = 20$ the same implementations have been done for the low-pass filter.

The implementations have been repeated for $h = 0, 2, 4, 6, 8, 10$. It is not useful to consider a truncated multiplier with $h > 10$, because the area occupation of the multiplier is very similar to full-width one.

5.2.2 Comparison with the analytical results

Applying 2^{17} samples of an input signal with uniform distribution to the low pass filter ($n = 16$) provides the results shown in Fig. 5.4. In the figure the simulation values are indicated with symbols while the theoretical values, computed with (5.11), with lines. The comparison confirms the correctness of the analytical calculation of Sec. 5.1.

The same samples are applied also to the high pass filter and to the band pass filter. The results, averaged on the three considered filters, are shown in Tab. 5.2. In the first and second column there are the mean error and mean square error. As example, for LMS1b multiplier with $h = 2$ the mean square error is $1.93 \cdot \mathbf{lsb}^2$ for low-pass filter, $1.10 \cdot \mathbf{lsb}^2$ for band-pass filter and $4.54 \cdot \mathbf{lsb}^2$ for high-pass filter, and the average value has been calculated as $2.52 \cdot \mathbf{lsb}^2$. The remaining columns show power dissipation, silicon area occupation and the maximum working frequency of the filter. For each value the percentage variation with respect to the value obtained using a full-width multiplier is also shown.

Tab. 5.2 demonstrates that the use of truncated multiplier with an output formed by the same number of bits of the input causes a very big mean error (as high as $100 \cdot \mathbf{lsb}$ on average for JOU [13]), even if silicon area can be reduced up to 31% while the working frequency increases up to 30%.

The best result in terms of mean square error is always provided by LMS1b truncated multiplier. In order to get an error lower than $\frac{1}{2}\mathbf{lsb}$ on the output,

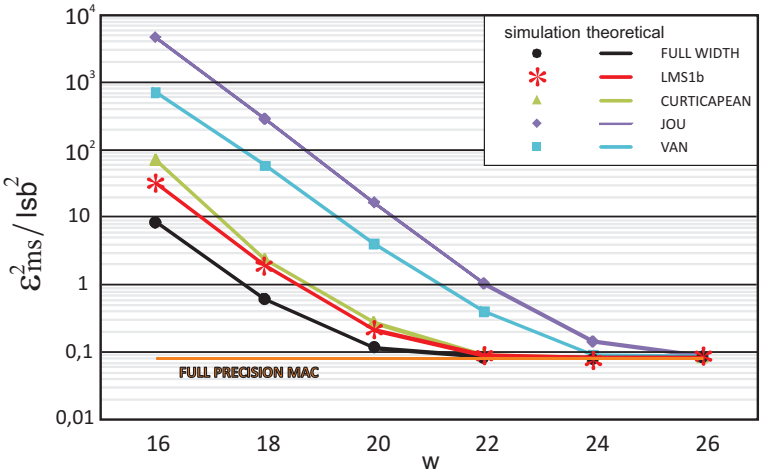

Figure 5.4: Mean square error of low pass filter implemented with different truncated multipliers varying the number of outputs bits w= $n+h$ ($n=16, h=0,2,4,6,8,10$). The applied input is a signal with uniform distribution. The lines represent the theoretical mean square error, obtained with (5.11), the symbols the simulation results. The theoretical value is experimentally verified. The orange line is the mean square error of the full precision MAC with a final rounding.

it is necessary to use a truncated multiplier with $h=4$. This solution for LMS1b multiplier provides satisfactory results in terms of area occupation (16% saving) and maximum working frequency (22% faster). The power dissipation reduction is 18%.
Only for $h=0$ LMS1b truncated multiplier is characterized by a slightly higher area and power dissipation (only 1%) but it has also the lowest mean square error. Finally the table shows that for $h \geq 6$ the (5.14) is verified for LMS1b and CCP [14] multipliers: their errors are equivalent to the reference, full-width, circuit. As a consequence these truncated multipliers (16 bits input and 20-22 bits outputs) can perfectly substitute a full-width multiplier, with a significant improvement in power, speed and area occupation.
As the reader can see in Tab. 5.3-5.4, this trend is still valid for $n=12$ and $n=20$. The best result in terms of mean square error is always provided by LMS1b truncated multiplier. Using a truncated multiplier with $h=4$ guarantees an error lower than $\frac{1}{2}\mathbf{lsb}$ on the output, with 12% ($n=12$) and

Table 5.2: Performances of the FIR Filters implemented using various Truncated Multipliers, with $n = 16$. The results have been obtained with an uniform distributed signal. Bold numbers indicate the best performing circuit for each *w(h)* value.

w		$\overline{\varepsilon_{FIR}}$	ε_{msFIR}	Power		Area		Frequency	
		(lsb)	($\mathbf{lsb}^2$)	($\mu W/MHz$)		($10^3\mu m^2$)		(MHz)	
(h)	Architecture	value	value	value	$\Delta\%$	value	$\Delta\%$	value	$\Delta\%$
16 (0)	Full-width	0.00	11.58	67.33		33.73		255	
	Jou[13]	-94.09	12702.40	45.94	-31.77	23.41	-30.58	332	**-30.23**
	Ccp[14]	9.13	166.07	48.76	-27.57	23.24	**-31.08**	303	18.79
	Van[15, 16]	35.74	1856.16	45.92	**-31.79**	23.40	-30.62	326	27.69
	LMS1b	**-1.18**	**40.61**	46.23	-31.33	23.46	-30.43	328	28.52
18 (2)	Full-width	0.00	0.80	74.58		39.41		253	
	Jou[13]	-23.48	794.04	58.48	-21.59	31.49	-20.08	284	12.22
	Ccp[14]	**0.15**	2.99	59.17	-20.65	31.47	-20.14	284	12.22
	Van[15, 16]	11.02	179.92	57.80	-22.50	31.26	-20.67	312	**23.05**
	LMS1b	-0.20	**2.52**	57.28	**-23.19**	31.24	**-20.73**	312	**23.05**
20 (4)	Full-width	0.00	0.13	77.08		40.11		252	
	Jou[13]	-5.77	48.61	65.30	-15.29	33.96	-15.33	296	17.46
	Ccp[14]	0.07	0.28	65.21	-15.40	33.93	-15.39	297	17.80
	Van[15, 16]	-	-	-	-	-	-	-	-
	LMS1b	**-0.06**	**0.23**	63.18	**-18.03**	33.67	**-16.04**	307	**21.78**
22 (6)	Full-width	0.00	0.09	79.36		40.97		251	
	Jou[13]	-1.42	3.06	70.45	-11.23	36.11	-11.87	299	19.46
	Ccp[14]	**0.00**	**0.09**	70.45	-11.23	36.09	-11.90	299	19.46
	Van[15, 16]	-	-	-	-	-	-	-	-
	LMS1b	**0.00**	**0.09**	69.02	**-13.03**	35.82	**-12.58**	303	**20.91**
24 (8)	Full-width	0.00	0.08	81.54		41.69		250	
	Jou[13]	-0.34	0.26	73.79	**-9.51**	37.99	-8.87	292	16.96
	Ccp[14]	**0.01**	**0.08**	73.93	-9.33	37.97	-8.91	292	16.96
	Van[15, 16]	-	-	-	-	-	-	-	-
	LMS1b	**0.01**	**0.08**	74.55	-8.57	37.66	**-9.66**	302	**20.85**
26 (10)	Full-width	0.00	0.08	83.09		42.34		250	
	Jou[13]	-0.08	0.09	77.83	**-6.33**	39.54	-6.61	292	16.96
	Ccp[14]	0.01	**0.08**	77.93	-6.20	39.53	-6.64	292	16.96
	Van[15, 16]	-	-	-	-	-	-	-	-
	LMS1b	**0.00**	**0.08**	77.91	-6.23	39.19	**-7.42**	299	**19.76**

Table 5.3: Performances of the Low-Pass Filters implemented using various Fixed-width Multipliers, for $n = 12$. The results have been obtained with a uniform distributed signal. Bold numbers indicate the best performing circuit for each *m(h)* value.

w (h)	Architecture	$\overline{\varepsilon_{FIR}}$ (lsb) value	ε_{msFIR} (lsb^2) value	Power ($\mu W/MHz$) value	Δ%	Area ($10^3\mu m^2$) value	Δ%	Frequency (MHz) value	Δ%
12 (0)	Full-width	0.00	7.33	38.34		20.72		324	
	Jou[13]	56.59	3216.08	28.94	-24.51	15.63	**-24.55**	379	17.05
	Ccp[14]	-7.47	68.76	30.24	-21.11	15.81	-23.70	353	9.19
	Van[15, 16]	-25.16	647.14	28.93	**-24.54**	15.62	-24.60	376	16.17
	LMS1b	**-0.76**	**21.28**	29.49	-23.07	15.66	-24.39	380	**17.49**
14 (2)	Full-width	0.00	0.56	47.32		26.49		281	
	Jou[13]	13.80	191.35	39.46	-16.60	22.31	-15.78	346	23.18
	Ccp[14]	**-0.42**	1.84	39.31	-16.92	22.30	-15.83	346	23.18
	Van[15, 16]	11.02	126.35	38.75	-18.11	22.16	-16.34	348	24.04
	LMS1b	-0.49	**1.58**	38.16	**-19.35**	22.13	**-16.45**	356	**26.69**
16 (4)	Full-width	0.00	0.12	49.30		27.12		279	
	Jou[13]	3.23	10.59	42.55	**-13.71**	24.07	-11.26	342	22.60
	Ccp[14]	0.07	0.28	42.85	-13.09	24.05	-11.32	342	22.60
	Van[15, 16]	-	-	-	-	-	-	-	-
	LMS1b	**-0.01**	**0.16**	43.00	-12.79	23.90	**-11.88**	347	**24.31**
18 (6)	Full-width	0.00	0.09	51.12		27.82		275	
	Jou[13]	0.72	0.60	46.00	**-10.02**	25.62	-7.90	330	19.80
	Ccp[14]	-0.03	**0.09**	46.32	-9.40	25.61	-7.95	330	19.80
	Van[15, 16]	-	-	-	-	-	-	-	-
	LMS1b	**-0.01**	**0.09**	46.20	-9.64	25.43	**-8.60**	332	**20.60**
20 (8)	Full-width	0.00	0.08	52.88		28.54		273	
	Jou[13]	0.14	0.10	49.05	**-7.24**	26.90	-5.75	326	19.22
	Ccp[14]	-0.02	**0.08**	49.41	-6.56	26.90	-5.77	326	19.22
	Van[15, 16]	-	-	-	-	-	-	-	-
	LMS1b	**0.00**	**0.08**	49.47	-6.44	26.69	**-6.49**	331	**21.19**
24 (10)	Full-width	0.00	0.08	54.28		29.26		272	
	Jou[13]	0.02	**0.08**	51.55	**-5.03**	27.90	-4.66	323	18.71
	Ccp[14]	0.02	**0.08**	51.55	**-5.03**	27.90	-4.66	323	18.71
	Van[15, 16]	-	-	-	-	-	-	-	-
	LMS1b	**0.00**	**0.08**	51.57	-4.73	27.89	**-4.70**	325	**19.48**

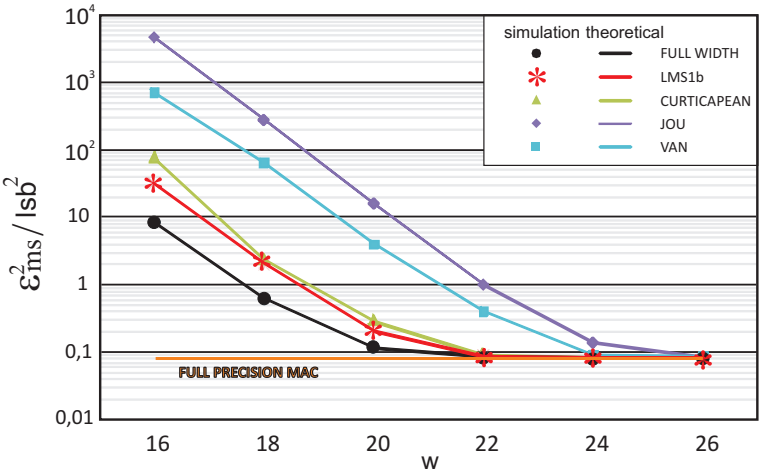

Figure 5.5: Mean square error of FIR implemented with different truncated multipliers varying the number of outputs bits $w = n + h$ ($n = 16, h = 0, 2, 4, 6, 8, 10$). The applied input is a sinusoid. The orange line is the mean square error of the full precision MAC with a final rounding.

17% ($n = 20$) saving in area occupation, 13% ($n = 12$) and 23% ($n = 20$) power dissipation reduction and a gain in frequency of 24% ($n = 12$) and 16% ($n = 20$).

5.2.3 Effect of the probability distribution of the input on the error

In Sec. 5.1 it has been demonstrated that the error depends on the pdf of the input, (5.12)-(5.13). In order to verify this equation and to analyze the behavior of the filter varying the inputs, every FIR filter considered in the previous section has been tested applying 2^{17} samples of a sinusoid and of two signals with different pdf: Gaussian (a signal with a rised cosine frequency spectrum) and Exponential. The results, averaged on the three considered filters, are shown in Fig. 5.5-5.6-5.7. The results are in good agreement with those obtained using (5.12)-(5.13). Fig. 5.5-5.6-5.7 show that the LMS1b multiplier has a very stable behavior as a function of the pdf of the input. The same happens to Van [15, 16] and JOU [13] multipliers. On the contrary, the CCP [14] multiplier

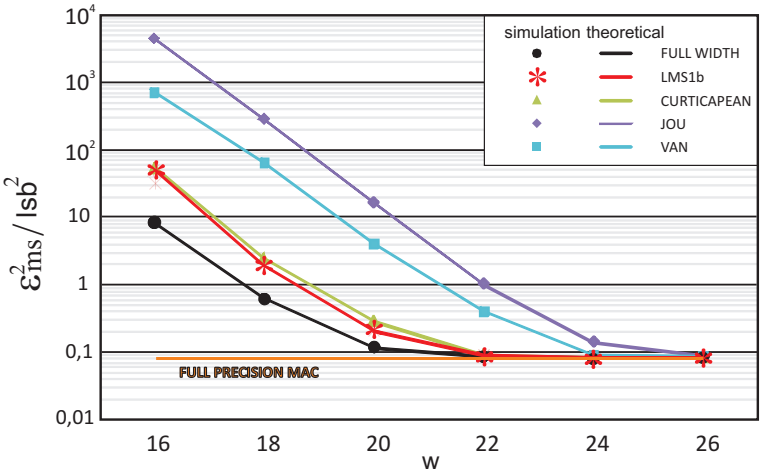

Figure 5.6: Mean square error of FIR implemented with different truncated multipliers varying the number of outputs bits $w = n + h$ ($n = 16, h = 0, 2, 4, 6, 8, 10$). The applied input has a gaussian probability distribution (rised cosine frequency spectrum). The orange line is the mean square error of the full precision MAC with a final rounding.

is more dependent on the applied input. The multiplier provides a high error for exponential distribution, a distribution for which the most probable input is formed by many bits equal to 1, and a low error for gaussian distribution, a distribution for which the most probable input is formed by many bits equal to 0.

5.2.4 Example of FIR filter

As example it has been implemented a 100 taps low-pass FIR filter, using the architecture of Fig. 5.3 with $w = n = 20$, $h = 2$. In this example the overflow prevention has been made with the use of 4 guard bits and the final rounding is realized by initializing the accumulator with the rounding constant before each accumulation. The sample frequency is assumed to be 48 MHz. The filter coefficients have been calculated with the equiripple algorithm by imposing a 1 MHz pass-band, a 1 MHz transition band and a 60dB attenuation in the stop-band. Fig. 5.8 shows the frequency response of the filter implemented with the architectures of Fig. 5.3, by using a LMS1b truncated multiplier with h=0. The response is compared with the response of the floating-point filter and with a

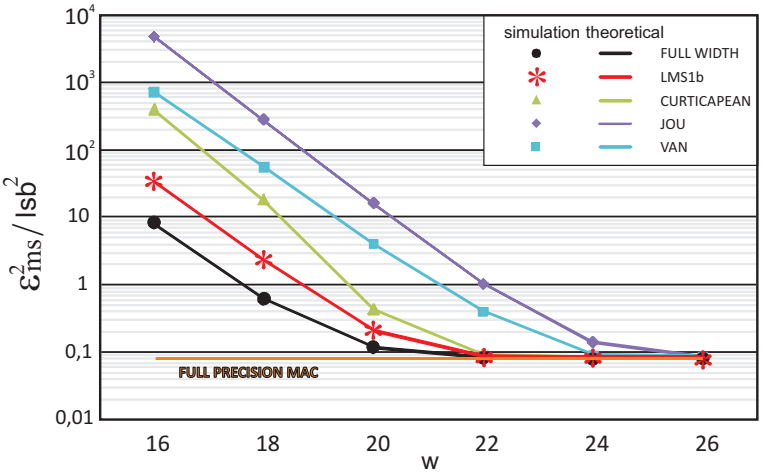

Figure 5.7: Mean square error of FIR implemented with different truncated multipliers varying the number of outputs bits $w = n + h$ ($n = 16, h = 0, 2, 4, 6, 8, 10$). The applied input has a exponential probability distribution. The orange line is the mean square error of the full precision MAC with a final rounding.

filter using a full-width (non-rounded) multiplier and a 36 bit accumulator. As can be seen the three frequency responses result almost indistinguishable.

5.3 Conclusion

The performances provided by different architectures of truncated multipliers when used to implement FIR filters using a single Multiply and Accumulate unit, has been investigated in this chapter. The error provided by the truncated multiplier on the output of the FIR is analytically calculated. The analytical equations show that in this kind of application it is important to use a truncated multiplier characterized by a small mean and mean-square error, for the given coefficients. In fact each result is given by the accumulation of a series of products, and hence positive and negative truncation errors have high probability to compensate each others. Various FIR filters have been implemented in TSMC 0.18μm technology, considering different truncated multipliers varying the number of output bits. The analysis shows that truncated multipliers are a suitable replacement for the full-width multipliers and that the optimal perfor-

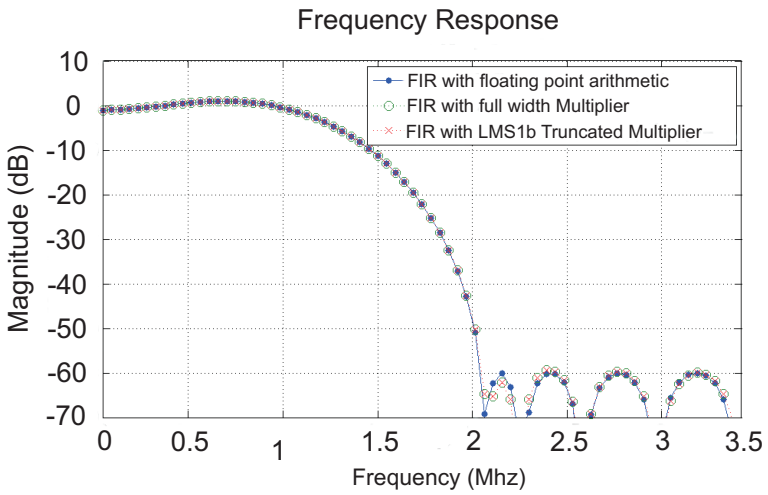

Figure 5.8: Frequency Response of the FIR filter implemented using the architecture of Fig. 5.2 and Fig. 5.3 with $w = n = 20$, $h = 2$.

mances are provided by the LMS1b proposed truncated multiplier able, for the proposed FIR filters, to keep the square error below $1/4\mathbf{lsb}^2$ while providing, for $n = 16$, a 16% and 18% reduction of the silicon area occupation and power dissipation with a 22% increase of the working frequency, and, for $n = 20$, a 17% and 23% reduction of the silicon area occupation and power dissipation with a 16% increase of the working frequency, while. Furthermore LMS1b multiplier is low sensible at the pdf of the input, and so the filter implemented with this multiplier can be used with various inputs, keeping the same mean square error.

Table 5.4: Performances of the Low-Pass Filters implemented using various Fixed-width Multipliers, for $n = 20$. The results have been obtained with a uniform distributed signal. Bold numbers indicate the best performing circuit for each *m(h)* value.

w		$\overline{\varepsilon_{FIR}}$	ε_{msFIR}	Power		Area		Frequency	
		(lsb)	(lsb^2)	($\mu W/MHz$)		($10^3\mu m^2$)		(MHz)	
(h)	Architecture	value	value	value	$\Delta\%$	value	$\Delta\%$	value	$\Delta\%$
	Full-width	0.00	7.33	79.70		53.59		183	
20	Jou[13]	72.59	5299.94	45.56	**-42.84**	37.38	**-30.26**	225	**23.20**
	Ccp[14]	4.32	56.38	51.63	-35.23	37.70	-29.65	207	12.02
(0)	Van[15, 16]	-26.17	715.50	49.04	-38.47	37.38	-30.25	220	20.22
	LMS1b	**-0.48**	**30.39**	47.16	-40.83	37.39	-30.23	220	20.48
	Full-width	0.00	0.56	90.61		66.65		182	
22	Jou[13]	17.85	320.73	63.47	-29.96	53.36	-19.94	205	12.70
	Ccp[14]	0.79	3.05	63.37	-30.06	53.33	-19.99	208	14.35
(2)	Van[15, 16]	-7.15	53.12	70.61	-22.07	53.04	-20.42	220	**20.88**
	LMS1b	**-0.34**	**2.90**	61.97	**-31.61**	52.99	**-20.49**	217	19.31
	Full-width	0.00	0.11	91.99		67.37		182	
24	Jou[13]	4.29	18.62	72.81	-20.85	56.41	-16.27	206	13.17
	Ccp[14]	**-0.25**	0.32	69.98	-23.92	56.37	-16.33	207	13.64
(4)	Van[15, 16]	-	-	-	-	-	-	-	-
	LMS1b	0.27	**0.30**	71.17	**-22.63**	56.01	**-16.87**	212	**16.53**
	Full-width	0.00	0.09	94.84		68.08		182	
26	Jou[13]	1.06	1.21	81.26	-11.66	59.12	-13.15	204	12.24
	Ccp[14]	**0.05**	**0.09**	81.33	-11.58	59.10	-13.19	204	12.24
(6)	Van[15, 16]	-	-	-	-	-	-	-	-
	LMS1b	**0.05**	**0.09**	77.62	**-15.62**	58.71	**-13.75**	211	**15.79**
	Full-width	0.00	0.08	97.47		68.78		182	
28	Jou[13]	0.28	0.16	83.46	**-14.37**	61.56	-10.49	203	11.79
	Ccp[14]	**0.01**	**0.08**	84.04	-13.78	61.54	-10.53	203	11.79
(8)	Van[15, 16]	-	-	-	-	-	-	-	-
	LMS1b	**0.01**	**0.08**	84.04	-13.78	61.12	**-11.13**	209	**15.06**
	Full-width	0.00	0.08	99.61		69.49		182	
30	Jou[13]	0.17	0.09	88.09	-11.57	63.68	-8.35	203	11.79
	Ccp[14]	0.01	**0.08**	88.19	-11.46	63.67	-8.37	203	11.79
(10)	Van[15, 16]	-	-	-	-	-	-	-	-
	LMS1b	**0.00**	**0.08**	87.45	**-12.21**	63.24	**-8.99**	207	**14.11**

Chapter 6

Temperature Control for Gas Sensors

Gas sensors are becoming increasingly important in our everyday lives. Their current applications include smoke alarms, laboratory analysis, medicine, automotive and industrial safety. However the range of applications is hinderer by the high cost (due to the semiautomated manufacturing methods) and high power consumption (due to the power needed to heat the sensing material and to reduce its response time) of currently available gas sensors [38].

Silicon-based micro gas sensors can overcome these problems. The small size helps in achieving low power consumption, while the use of existing microelectronics technology can greatly reduce manufacturing costs.

New silicon-based gas sensors using micro-hotplate technology are in great need of accurate temperature control. In fact the accurate determination of the working temperature allow to determine the gas that is detected and precisely monitors its concentration. In this chapter the implementation and the performance of two controllers, On/Off and PI will be described. Using the proposed truncated multiplier the PI controller will be optimized and it will be compared also with a new controller, based on a mixed PI-On/Off approach.

6.1 Resistive gas sensors

Nowadays most of the resistive gas sensors use metal oxides for the sensing material. Metal oxides (e.g. zinc oxide [39]) react with different gases at fairly high temperatures (300°C- 500°C) when their resistance changes as function

of the gas concentration. Therefore, from the change in resistance and the optimum temperature of operation it is possible to figure out the concentration and nature of the gas. However, the high working temperature demands very large power, e.g. the most widely available resistive gas sensor (Taguchi type, sold by Figaro, Japan) has an operating temperature of typically $400°C$ and power consumption of between 500mW and 800mW [40]. Hence, researchers have been trying to reduce the power consumption by changing the technology and the design of the gas sensor device.

One of these approaches is a membrane based gas sensor [41]. This approach is particularly useful when it is integrated with CMOS, since on-chip read out circuits can be used. The membrane structure contains a micro-heater (to heat up the sensing material), the interdigitated electrodes (to measure the change in resistance of the sensing material) and a temperature sensor (to measure accurately the temperature of the sensitive material). The membrane approach is an innovative approach with which it is possible to achieve lower power consumption by thermal isolation of the micro-heater from the rest of the chip through the removal of the bulk silicon (by deep reactive ion etching or KOH etching). A further reduction of the power dissipation is obtained using a fast pulse mode operation of the micro-heater (instead of providing a constant supply all the time). In this case an accurate temperature control of the micro-heater is particularly important to maintain the desired temperature in the sensing layer.

Several designs of high temperature micro gas sensors can be found in literature [42, 43, 44]. Here no more details will be given since in this chapter the attention is focused on the temperature control.

The resistive gas sensor used in this chapter is presented in [45]. It is a tungsten based microhotplate fabricated in a commercial silicon on insulator (SOI)-CMOS process. Tungsten, used as metallization in some commercial SOI processes (for increasing the junction temperature of the CMOS), has good mechanical strength and can operate reliably at high temperatures. Thanks to the reduced power dissipation given by the SOI technology, it is possible to heat the sensing material up to $450°C$ in a short time. The structure is shown in Fig. 6.1. During the fabrication of the interface circuitry, the tungsten metal layers (used as interconnect in CMOS circuitry) are used to form the heater, a heat spreading plate, and interdigitated electrodes for gas sensing. Back etching by deep reactive ion etching (DRIE) is then used to release the membrane, followed by the deposition of the gas sensing layer.

The layout of the micro-hotplate device is shown in Fig. 6.2. The heater

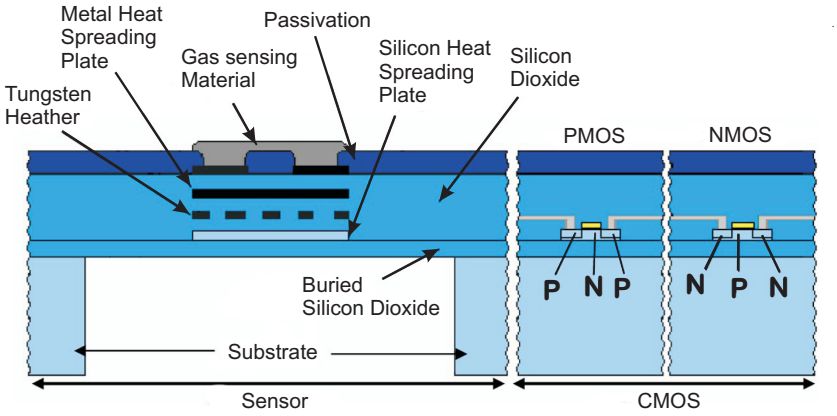

Figure 6.1: Tungsten SOI chip: gas sensor and integrated CMOS circuitry.

is rectangular ring shaped and heats an area of $185\mu m \times 185\mu m$. Interdigitated electrodes have been designed within the heater area where the sensing material is deposited. There is also a bipolar transistor temperature sensor (implemented as a diode) below the heater and this is used by the controller to measure the temperature of the membrane. The membrane area used for this particular sensor was $530\mu \times 530\mu m$. The device has been fabricated in a commercial CMOS foundry, followed by a separate wet etching process (to form the membrane) in MEMS foundry

6.1.1 Interface Circuitry

The interface circuitry consists of driving circuits for heater and temperature sensor and the On/Off circuitry to control the temperature of the heater. The micro-heater can be driven in a voltage or a current control mode. In case of voltage mode, one needs to use current limiting resistor in series with micro-heater in order to increase the lifetime of the heater [46], consequently, current drive circuitry is preferred and has been selected for the purpose of this work.

A cascode current mirror was designed to drive 20mA maximum current through the heater. The temperature sensor, bipolar junction transistor with base and collector terminal shorted together, was driven by a constant current (of $\sim 65\mu A$). The voltage across the temperature sensor reduces linearly with increase in temperature (0.87 V @ room temperature to 0.45 V @ 375 °C, i.e. -1.2 mV/°C). The detailed discussion on the temperature sensor performance

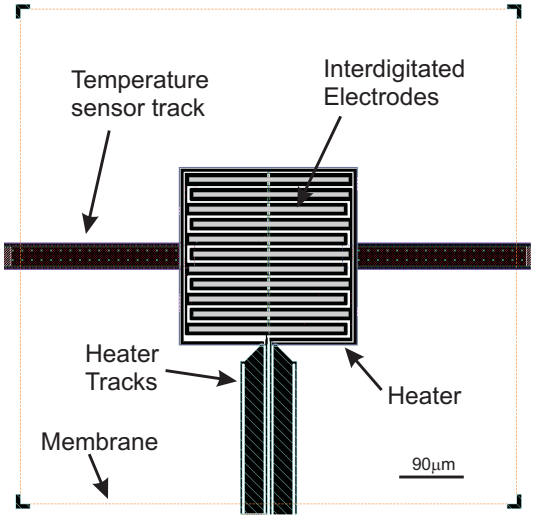

Figure 6.2: Layout of the microhotplate device.

was reported elsewhere [47]. Circuit schematic is shown in Fig. 6.3, it also shows the A/D converter and the Temperature controller.

6.2 Temperature Control

It is very important to accurately control the temperature of the sensing layer. Often due to variations in the ambient temperature, metal electromigration in the heater or the simple variability in the dimensions of the heater during fabrication lead to significant variations in the power vs. temperature characteristic. As a consequence it is not possible to rely on power measurement to estimate the temperature of the sensitive layer. The only way to accurately control the temperature of the sensing layer is using an accurate temperature sensor embedded in the micro-hotplate in conjunction with a CMOS controller to detect and adjust sensing layer temperature.

Several classes of controllers have been implemented with digital CMOS circuitry due to the great advantages that the digital circuits provide over their analogue counterparts. They are less expensive, more reliable, easy to manipulate and flexible. As it is shown in Fig. 6.3, a temperature sensor reads the temperature of the microhotplate. This value $y(t)$ is converted into a digital form

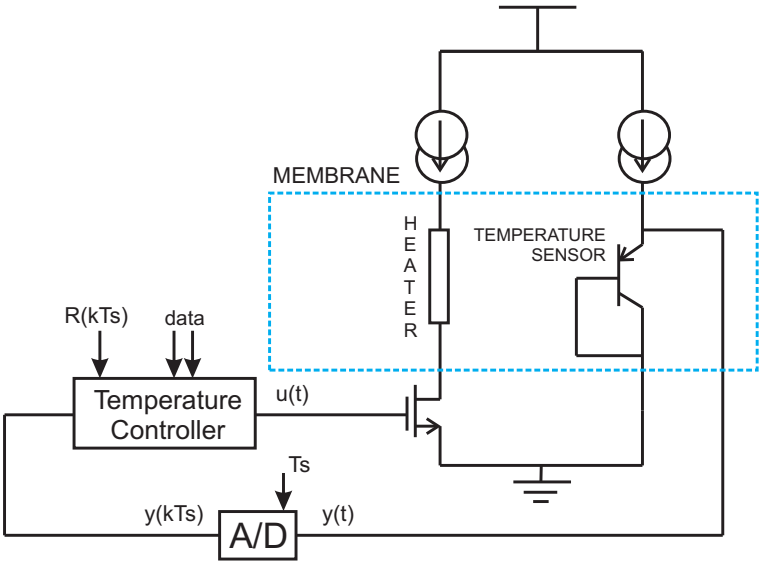

Figure 6.3: Schematic of the circuit composed by the gas sensor (membrane), cascode current mirror to drive heater and temperature sensor, A/D converter and a temperature controller. $y(t)$ is the measured temperature, $R(kT_S)$ the desired temperature, $u(t)$ is the control variable and T_S is the sampling period.

and compared with the desired value of temperature ($R(kT_S)$), where T_S is the sampling period. The controller takes this error $e(kT_S) = y(kT_S) - R(kT_S)$ and, depending on the implemented controlling function, drives the microhotplate in order to reach the desired temperature.

6.2.1 On/Off control

An On/Off controller is the simplest form of a temperature control device. The output from the device is either on or off, with no middle state. When the microhotplate is cooler than the set-point temperature the heater is turned on at maximum power, and once it is hotter than the set-point temperature the heater is switched off completely. The On/Off control is designed to include hysteresis: there is a deadband, a region around the setpoint value in which no control action is needed. The width of the deadband is adjustable from outside

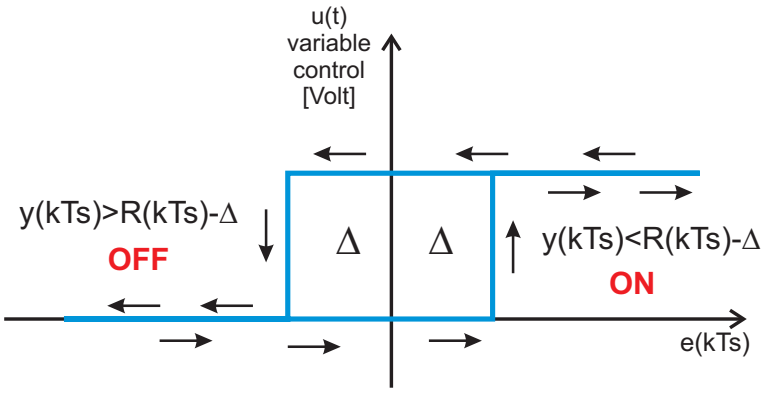

Figure 6.4: On/Off controller.

(see Fig. 6.4).

The working principle of the On/Off controller can be described with the following equation. The output of the controller at $k+1^{th}$ instant will be:

$$u\left(\left(k+1\right)T_S\right) = \begin{cases} 0 \quad if \begin{cases} u\left(kT_S\right) = 1 \\ e\left(kT_S\right) < -\Delta \end{cases} \\ 1 \quad if \begin{cases} u\left(kT_S\right) = 0 \\ e\left(kT_S\right) > \Delta \end{cases} \\ u\left(kT_S\right) \quad \text{otherwise} \end{cases} \tag{6.1}$$

where:

1. T_S is the sampling period;
2. $e(kT_S)$ is the difference between the setpoint $R(kT_S)$ (desired temperature) and the measured temperature $y(kT_S)$
3. $u(kT_S)$ the output of the controller at the *h*-th instant
4. Δ the width of the deadband

On/Off control is very simple and cheap, but persistent oscillation of the process variable occurs and there is a continuous cycling of the controlled variable and excess wear on the final control element. Hence it is usually used where a precise control is not necessary or in systems which cannot handle

frequent On/Off switches or in those cases in which system temperature change extremely slowly.

6.2.2 PI control

Proportional controls are designed to eliminate the cycling associated with the On/Off control. A proportional controller decreases the average power supplied to the heater as the temperature approaches the setpoint. This has the effect of slowing down the heater so that it will not overshoot the setpoint, but will approach the setpoint and maintain a stable temperature. In order to reduce to zero the steady-state error produced by the proportional action (rejecting the offset), an integral action is also applied. This second term is proportional to the integral of the error. So the control output is given by:

$$u(t) = K_P \left(e(t) + \frac{1}{T_i} \int_0^t e(\tau)\, d\tau \right) \tag{6.2}$$

where e(t) is the difference between the desired and the measured value of temperature, K_P is the proportional gain and T_i the integral time. The ideal PI controller has been digitalized and his output is used by a PWM block to vary the desired percent of duty cycle of a signal applied to heather. PWM is a Pulse Width Modulation in which the duty cycle of a square wave is modulated to encode a specific analog signal level: the voltage source is supplied to the analog load by means of a repeating series of on and off pulses. The on-time is the time during which the DC supply is applied to the load, the off-time is the period during which that supply is switched off. Hence the heater will be driven with a mean voltage equal to the ones needed to keep the desired value of temperature.

In order to formulate a discrete PI controller, we apply the backward difference methods to the expression of the system in the Laplace domain, obtaining:

$$u(kT_S) = u(kT_S - T_S) + K_P \left(e(kT_S) - e(kT_S - T_S) \right) + \frac{K_P T_S}{T_i} e(kT_S) \tag{6.3}$$

Finally the output is given by:

$$u(n) = u(n-1) + a \cdot e(n) + b \cdot e(n-1) \tag{6.4}$$

where $n = kT_S$.

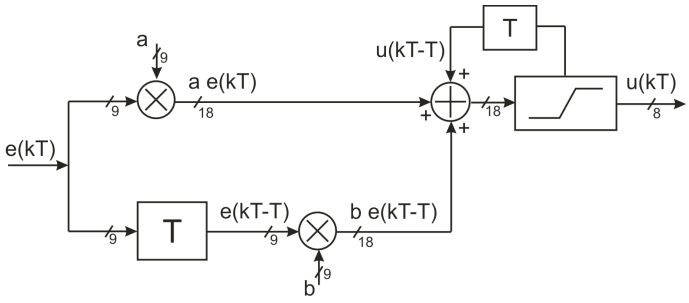

Figure 6.5: Digital implementation of PI algorithm.

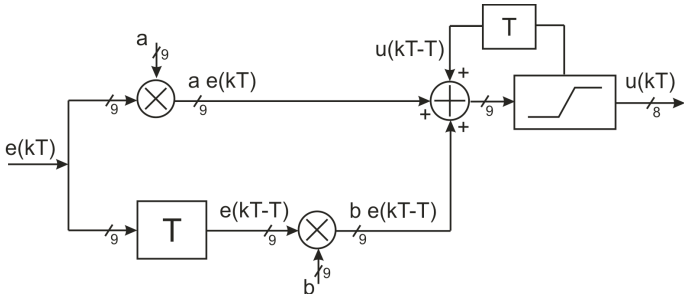

Figure 6.6: Digital implementation of an optimized PI algorithm.

This expression is also known as *velocity PI algorithm*: the calculation of current control uses the previous control value as reference, hence the control is calculated as a change. The digital implementation of the PI controller is represented in Fig. 6.5. As it is shown in this figure, there are two registers, a 9 bits register which stores the previous value of the error and a 18 bits register which keeps the previous value of the control variable, and a saturation block, with which the overflow of the control variable is prevented (as at each clock cycle a positive quantity could be added to previous value). Finally, with a rounding operation, the output is converted from 18 to 8 bits, in order to have the proper signal that should be applied to the PWM block.

In the circuit of Fig. 6.5 most of area consumption is determined by the two multipliers. Hence it can be useful to use the proposed LMS1b truncated multiplier for the implementation of the PI. Using this truncated multiplier an optimized PI controller has been implemented. The final circuit is represented

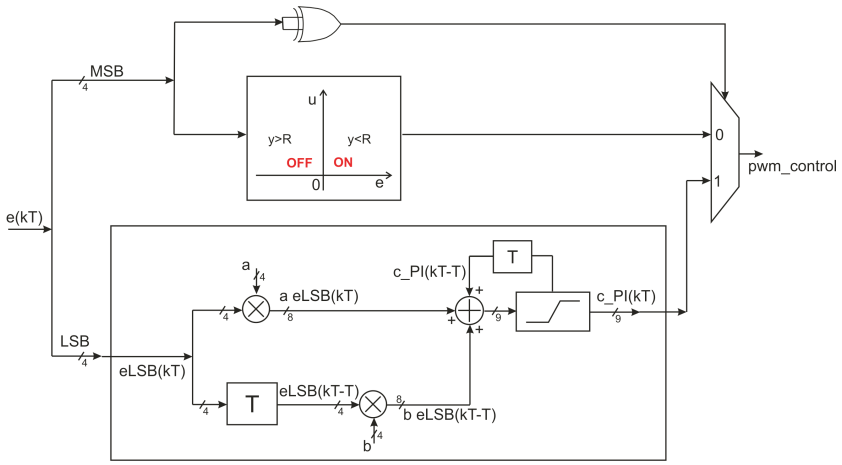

Figure 6.7: Mixed controller.

in Fig. 6.6. As it is shown in this figure the output of the multiplication is on the same number of bits of the inputs. Therefore, as already mentioned, the area occupation is reduced because of the use of this fixed-width multiplier, and furthermore the second register is formed by only 9 bits, as it will be shown in Sec. 6.3.

6.2.3 Mixed control

PI controllers allow a better adjustment in the system compared to On/Off controller, leading to a more accurate control of the variable. But there are some advantages to using an On/Off controller instead of a PI controller. The On/Off controller is cheap, both the design and the operation principle are simple. Nevertheless it is inefficient, it can generate noise (because it can dramatically overshoot or undershoot a set-point), and gives low accuracy on the control. Hence a good solution could be to use a mixed control.

The idea of MIX control is to apply the best possible control depending on the current value of the error. If the error is very high it is not necessary to apply a PI control to the heater. We can simply switch it on or off depending on the sign of the error. If the error is positive (the desired value is higher than the measured value) the heather will switch on, otherwise it will switch off. On the opposite, when the value of the error becomes small, we can apply a more accurate control, such as a PI control. The final circuit is shown in Fig. 6.7.

The error is split into two parts. The most significant part (MSP), formed by the first 5 bits of the errors, and the less significant parts (LSP), formed by the last 4 bits.

If every bit of the MSP part are 0 or to 1, the absolute value of the error is smaller than a small quantity, so we can apply a PI control. In this case we only need a multiplier formed by 4 bits, because we know the other part of the multiplication (formed by all 0 or all 1). Otherwise we can directly apply an On/Off control. Depending on the sign of the error we will switch the heather on or off. Finally a multiplexer is present in order to select which control we will apply.

6.3 Silicon Implementation

The proposed controller was designed in Verilog, synthesized using RTL compiler and than the layout of the final chip $2mm \times 3mm$ was generated with SoC Encounter. The design has been carried out in $0.8\mu m$ CMOS process with three levels of metal. The chip layout is shown in Fig. 6.8. It is possible to identify the Mixed controller, the PI controller, the Optimized PI controller and the On/Off controller (implemented using two or three levels of metal). The differences in terms of area occupation between them controller are visible. The amount of these differences can be seen Tab. 6.1, where also the other hardware characteristic are reported. The On/Off controller provides the minimum area occupation and power dissipation and maximum working frequency. If however, the most important requirement is the accuracy, one should use a PI controller. The problem is that it will occupy an area five times larger than that of the On/Off controllers.

By using a truncated multiplier the same accuracy is obtained and the area occupation and power dissipation of the PI controller can be reduced by as much as 50%

The best compromise between accuracy and physical implementation is provided by the novel MIXED Controller. It is quite rough if the error is very big but very precise when the error become smaller and smaller. The Area consumption is only double that of the On/Off ones and also the power dissipation is relatively low.

Since having a precise temperature control is important for accurate gas detection, it is very useful to have this control on-chip. Fig. 6.9 shows the layout of the microhotplate integrated with one of the analyzed controller, the On/Off. In the figure it is possible to see the analog driver circuit, the micro-

Table 6.1: Comparison Between Controllers.

Controller	Frequency [MHz]	Power [$\mu m/MHz$]	Area [mm^2]
On/Off Controller	60	157	0.08
PI Controller	22	825	0.41
Optimized PI Controller	25	340	0.26
Mixed Controller	25	390	0.18

heater, the On/Off controller and the A/D converter. The design has the flexibility of changing deadband region from outside the chip. The ability to define the frequency and the deadband allow a simple means to digitally control the accuracy of the micro-hotplate temperature, hence the gas detection.

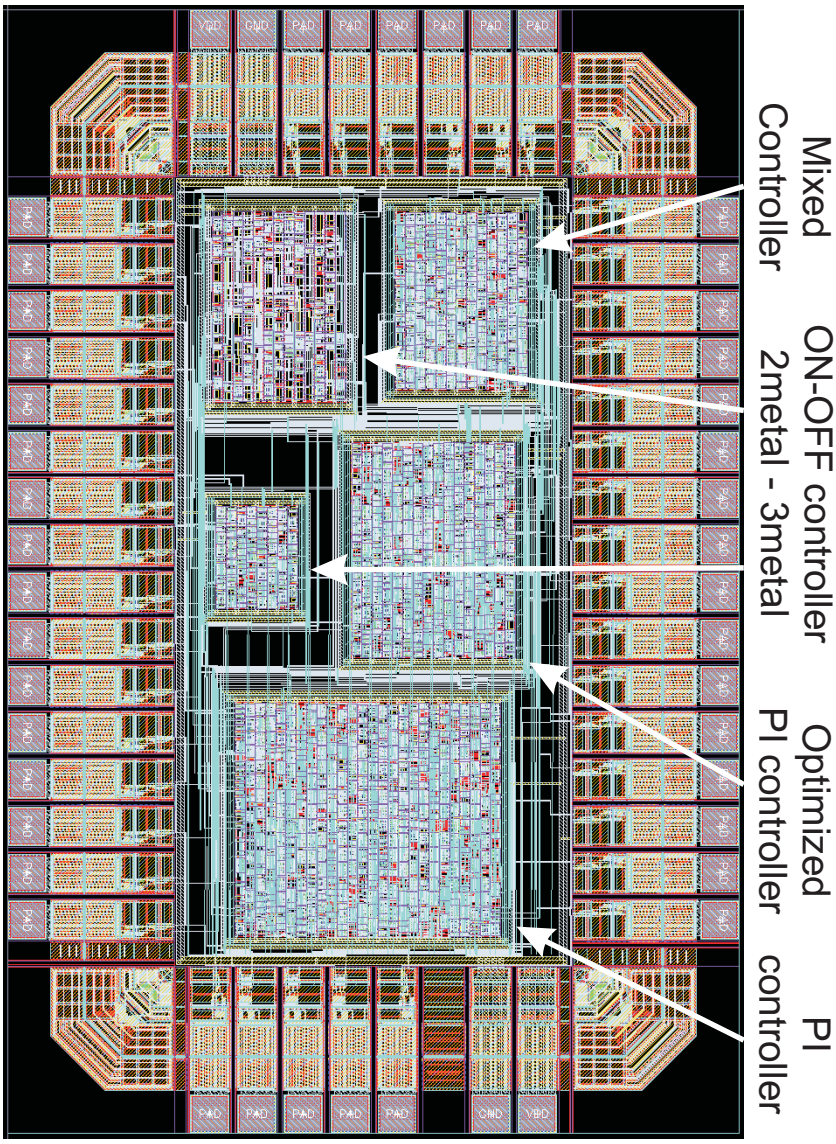

Figure 6.8: Digital chip with controllers.

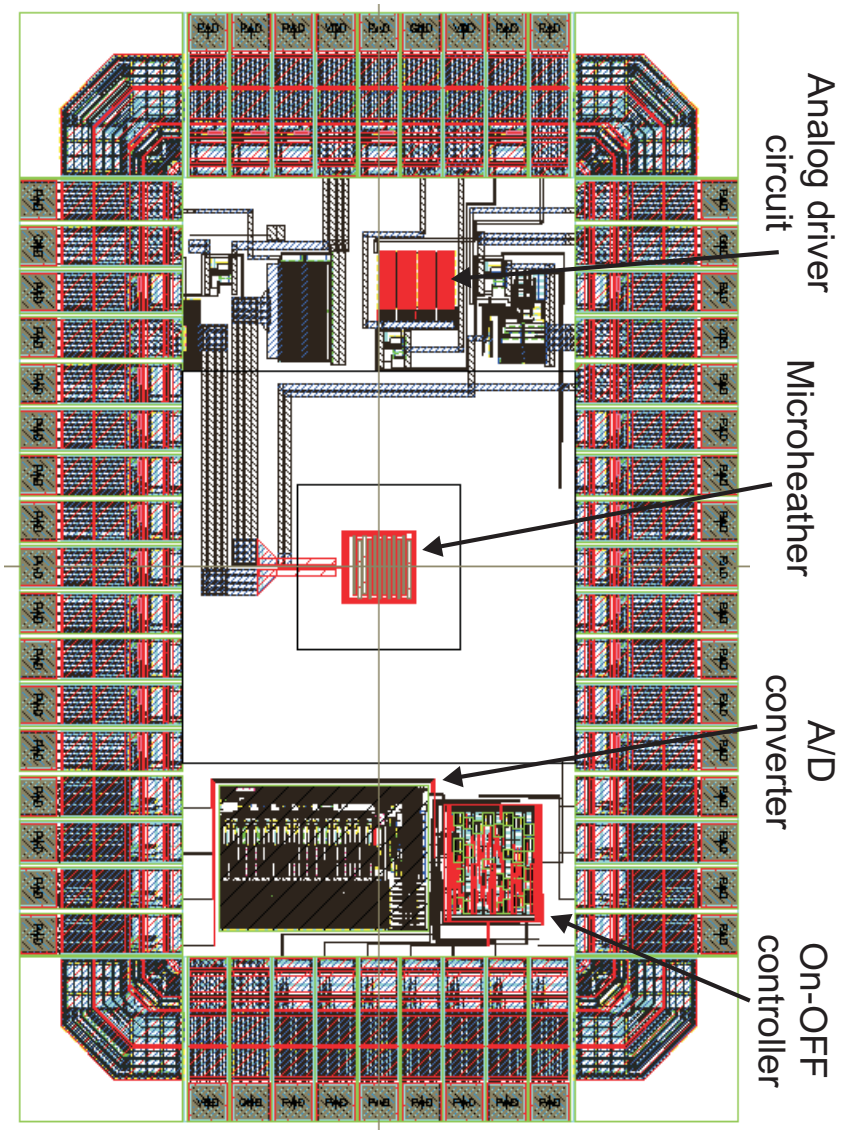

Figure 6.9: Mixed signal chip.

Conclusions

This dissertation has presented methods for the design of truncated multipliers and squares that allows to achieve area, delay and power benefits with the minimum mean square error.

The state of the art truncated multipliers show substantial limits. In particular, they are not derived from an analytical theory but rather heuristically or with the help of exhaustive searches. Thus, in many cases the proposed techniques can not be applied to multipliers with large bit widths (say, 24 or 32 bits) and/or can not be considered for a possible implementation in automatic synthesis tools. In addition, the errors are computed numerically through exhaustive simulations. This approach can be pursued only for small n values since the simulation time increases as O(2^{2n}), requiring an unreasonable amount of CPU time when n increases.

On the contrary the approach proposed in this dissertation is analytical. Looking at the error due to the approximation of some bits of the result with a compensation function, the optimal compensation function, which minimize the mean square error, has been computed.

The first result is that the optimal compensation function is a quadratic form of the partial products of the IC; this is due to the approximation of a matrix with a vector.

A second important result is the evaluation, in closed form, of the intrinsic error, the mean square error introduced by the optimal compensation function. Until now it has been thought that by using a suitable compensation function the lowest error that could be obtained was the rounding error; indeed it has been demonstrated that the lowest error is represented by $\varepsilon^2_{\mathrm{low\_bound}}$, independent by the chosen compensation function. The analytical relation between this error and the parameters of hardware implementation gives to the designer the minimum characteristic that the truncated multiplier should have in order to give the desired accuracy.

The optimal compensation function, being a quadratic form, cannot be ef-

ficiently implemented in hardware. Therefore the performances achievable by using a linear compensation function, best suited for hardware implementation, have been investigated. It is shown that the additional error due to the linear compensation function is negligible, pointing out that this solution is the best choice from a practical point of view. The effect of coefficient quantization is also treated by providing the quantized optimal coefficients, using two different levels of quantization, given by only one bit (LMS1b truncated multiplier) and two bits (LMS2b truncated multiplier).

The mean square error and the maximum absolute error of the LMS truncated multiplier have been computed in closed form. This is one of the most important characteristic of the proposed LSM multiplier. It is the only architecture that can be designed, for every bit width, using an analytical approach that allows the a priori knowledge of the committed error. Without the analytical solution, it can only be computed using slow exhaustive simulations, possible only for low n values.

The analysis of the practical implementation of truncated multipliers with the quantized linear compensation function has been done and it has been demonstrated that the implementation of LMS1b multiplier is straightforward, while the optimal implementation of LMS2b multipliers is more challenging. The best approach uses a small auxiliary carry-save tree to minimize the hardware. The comparison of performances of the truncated multipliers developed in this thesis with previously proposed circuits has demonstrated that the proposed solution is the one which provides the best trade-off between hardware and accuracy.

All the results have been extended to the truncated binary squarer. Also in this case it has been obtained the analytical solution for the optimal compensation function, which minimizes the mean square error, and its linear approximation. The analytical technique results in a very simple circuit. Compared with other truncated squarer circuits proposed in the literature, the LMS squarer reduces the mean square error, the power dissipation, the silicon area occupation, and increases the maximum working frequency.

The error provided by the truncated multiplier on the output of the FIR filter has been analytically calculated. The analytical equations show that in this kind of application it is important to use a truncated multiplier characterized by a small mean and mean-square error, for the given coefficients.

Various FIR filters have been implemented in TSMC 0.18μm technology, considering different truncated multipliers varying the number of output bits. The analysis shows that truncated multipliers are a suitable replacement for

the full-width multipliers and that the optimal performances are provided by the proposed LMS1b truncated multiplier. Furthermore LMS1b multiplier is low sensible at the pdf of the input, and so the filter implemented with this multiplier can be used with various inputs, keeping the same mean square error.

Finally it has been demonstrated that the use of truncated multiplier can be useful also in others field, for example in the implementation of the PI temperature control. By using the truncated multiplier is possible to save area and power, even if the accuracy is a bit low. Note that, in order to obtain a better accuracy, the coefficients of the controller can be adjusted (automatic tuning is also possible).

Appendix A

Intrinsic Error

A.1 Computing $\sigma^2_{i,j}$

By using (2.47) in (2.55) one obtain:

$$\sigma^2_{ij}(A) = \mu_{ij}(A) - \mu^2_{ij}(A) = \mu_{ij}(A)\left[1 - \mu_{ij}(A)\right] = \\ = \frac{1}{9}\left(1 + 2\gamma_{i-h}\right)\left(1 + 2\gamma_{n+1-j}\right)\left[1 - \frac{1}{9}\left(1 + 2\gamma_{i-h}\right)\left(1 + 2\gamma_{n+1-j}\right)\right] \tag{A.1}$$

Developing the product in (A.1) and remembering that $\gamma_i^2 = \gamma_i$, one obtains:

$$\sigma^2_{ij}(A) = \frac{1}{81}\left[8 + 10\gamma_{i-h} + 10\gamma_{n+1-j} - 28\gamma_{i-h}\gamma_{n+1-j}\right] \tag{A.2}$$

A.2 Computing the covariance $\mathrm{COV}_{i,j,l,m}(A)$

In order to compute the covariance (2.56) in explicit form, we need to compute the mean of the product $x_i y_j \cdot x_l y_m$ in every set $\Omega(A)$. $x_i y_j$ and $x_l y_m$, correlated to γ_{i-h}, γ_{n+1-j} and γ_{l-h}, γ_{n+1-m} respectively, are conditionally independent for each fixed value of IC (therefore fixed the value that bits of the product can assume). Hence:

$$\underset{IC=A}{E}\left\{x_i y_j \cdot x_l y_m\right\} = \underset{IC=A}{E}\left\{x_i\right\} \cdot \underset{IC=A}{E}\left\{y_j\right\} \cdot \underset{IC=A}{E}\left\{x_l\right\} \cdot \underset{IC=A}{E}\left\{y_m\right\} \tag{A.3}$$

where the conditioned mean is equal to 1 if the correlated element of IC is equal to 1, $1/3$ otherwise (see (2.47)). Five different cases must to be considered.

Case 1: j=m This condition happens when the two partial products $x_i y_j$ and $x_l y_m$ belong to the same row of the $\text{LSP}_{\text{minor}}$. Hence the product $x_i y_j x_l y_j = x_i y_j x_l$ is correlated to three elements of IC: γ_{i-h}, γ_{n+1-j} and γ_{l-h}. Applying (A.3), the mean of the product is:

$$\underset{IC=A}{E}\{x_i y_j \cdot x_l y_j\} = \frac{1}{27} \cdot (1 + 2\gamma_{i-h}) \cdot (1 + 2\gamma_{n+1-j}) \cdot (1 + 2\gamma_{l-h}) \quad \text{(A.4)}$$

$$\underset{IC=A}{E}\{x_i y_j x_l\} \in \left\{1, \frac{1}{3}, \frac{1}{9}, \frac{1}{27}\right\}$$

Case 2: i=l This is a dual case with respect to the case 1, when the two partial products belong to the same diagonal of $\text{LSP}_{\text{minor}}$. The generic element $x_i y_j x_i y_m = x_i y_j y_m$ is correlated to three elements of IC: γ_{i-h}, γ_{n+1-j} and γ_{n+1-m}. Applying (A.3), the mean of the product is:

$$\underset{IC=A}{E}\{x_i y_j \cdot x_i y_m\} = \frac{1}{27}(1 + 2\gamma_{i-h}) \cdot (1 + 2\gamma_{n+1-j}) \cdot (1 + 2\gamma_{n+1-m}) \quad \text{(A.5)}$$

$$\underset{IC=A}{E}\{x_i y_j y_m\} \in \left\{1, \frac{1}{3}, \frac{1}{9}, \frac{1}{27}\right\}$$

Case 3: m+i=n+h+1 In this case $x_i y_m = x_i y_{n+h+1-i}$ is an element of the IC. Hence it is fixed in each set $\Omega(A)$:

$$\begin{aligned}\underset{IC=A}{E}\{x_i y_j \cdot x_l y_m\} = x_i y_m \underset{IC=A}{E}\{y_j \cdot x_l\} = \gamma_{i-h} \cdot \mu_{l,j} = \\ = \gamma_{i-h}\frac{1}{9}(1 + 2\gamma_{l-h})(1 + 2\gamma_{n+1-j}) \quad \text{(A.6)}\end{aligned}$$

Case 4: l+j=n+h+1 This is the dual case of the case 3:

$$\begin{aligned}\underset{IC=A}{E}\{x_i y_j \cdot x_l y_m\} = x_{n+h+1-j} y_j \underset{IC=A}{E}\{x_i \cdot y_m\} = \gamma_{n+1-j} \cdot \mu_{i,m} = \\ = \gamma_{n+1-j}\frac{1}{9}(1 + 2\gamma_{i-h})(1 + 2\gamma_{n+1-m}) \quad \text{(A.7)}\end{aligned}$$

Case 5: uncorrelated partial products When no previous case is verified, the two partial products are uncorrelated and hence the mean of their product is equal to the product of their means:

$$\underset{IC=A}{E}\{x_i y_j \cdot x_l y_m\} = \mu_{i,j} \cdot \mu_{l,m} \quad \text{(A.8)}$$

A.3 Computing the intrinsic error

In order to simplify the reasoning, the second (2.10) can be rearranged as follows:

$$\sum_{i=1}^{n-h-1}\sum_{j=2}^{n-h-i+1} x_{i+j-1+h}y_{n+1-i}2^{-n-h-j} \tag{A.9}$$

$\varepsilon^2_{\text{intrinsic}}$ is given by two contribution $S_{\sigma ij}$ and $S_{\text{COV}ijlm}$.

As told in Sec. 2.3.1 $S_{\sigma ij}$ is the sum of the variances of each partial product $\sigma^2_{i,j}(A)$ (multiplied by the square of partial product weight):

$$S_{\sigma ij} = \sum_{i=1}^{n-h-1}\sum_{j=2}^{n-h-i+1} \sigma^2_{i+j-1+h,n+1-i}(A)2^{-2h-2n+1} \tag{A.10}$$

substituting (A.2) in (A.10) one obtains:

$$S_{\sigma ij} = \sum_{i=1}^{n-h-1}\sum_{j=2}^{n-h-i+1} \frac{1}{81}\left[8+10\gamma_{i+j-1}+10\gamma_i-28\gamma_{i+j-1}\gamma_i\right]2^{-2h-2n+1} \tag{A.11}$$

$S_{\text{COV}ijlm}$ is the sum of twice the covariance between every couple of different partial products, $\text{COV}_{i,j,l,m}(A)$, (multiplied by the products of their weights). $S_{\text{COV}ijlm}$ can be divided in three terms, in order to highlight the single addends discussed previously.
The first term, T_1, is given by the covariance of the terms on the same row, as computed in (A.4):

$$\begin{aligned} T_1 = 2^{-2h-2n+1}\sum_{i=1}^{n-h-2}\sum_{j=2}^{n-h-i}\sum_{m=j+1}^{n-h-i+1} 2^{-j-m}\frac{2}{81}(1-\gamma_i+ \\ +2\gamma_{i+j-1}+2\gamma_{i+m-1}-2\gamma_i\gamma_{i+j-1}-2\gamma_i\gamma_{i+m-1}+ \\ +4\gamma_{i+j-1}\gamma_{i+m-1}-4\gamma_i\gamma_{i+j-1}\gamma_{i+m-1}) \end{aligned} \tag{A.12}$$

The second term, T_2, is given by the covariance of the terms discussed in the case 2 (A.5):

$$\begin{aligned} T_2 = 2^{-2h-2n+1}\sum_{i=1}^{n-h-2}\sum_{j=3}^{n-h+1-i}\sum_{m=i+1}^{i+j-2} 2^{-i-2j+m}\frac{2}{81}(1-\gamma_{i+j-1}+2\gamma_i+ \\ +2\gamma_m-2\gamma_{i+j-1}\gamma_i-2\gamma_{i+j-1}\gamma_m+4\gamma_i\gamma_m-4\gamma_i\gamma_{i+j-1}\gamma_m) \end{aligned} \tag{A.13}$$

The final term, T_3, is given by both the terms discussed in the case 3 and in the case 4 whose reference equations are (A.6) and (A.7):

$$T_3 = 2^{-2h-2n+1} \sum_{i=1}^{n-h-2} \sum_{j=2}^{n-h-i} \sum_{m=2}^{n-h-i-j+2} \frac{2^{-j-m}}{81} (-1 + \gamma_{i+j-1} - \\ - 2\gamma_i - 2\gamma_{i+j+m-2} + 2\gamma_{i+j-1}\gamma_i + 2\gamma_{i+j-1}\gamma_{i+j+m-2} - \\ - 4\gamma_i\gamma_{i+j+m-2} + 4\gamma_i\gamma_{i+j+m-2}\gamma_{i+j-1}) \tag{A.14}$$

Finally the $\varepsilon^2_{\text{intrinsic}}$ is given by:

$$\varepsilon^2_{\text{intrinsic}} = \sum_{A \in \Theta} (S_{\sigma ij} + T_1 + T_2 + T_3)\, P(A) \tag{A.15}$$

In order to solve the eq. (A.15) we can invert the external sum over the set Θ with the internal sums on indexes *i*, *j*, and *m*. Furthermore as

$$\sum_{A \in \Theta} \gamma_i P(A) = \sum_{A \in \Theta} x_{h+i} y_{n+1-i} P(A) = \sum_{A \in \Theta} x_{h+i} P(A) \sum_{A \in \Theta} y_{n+1-i} P(A) \\ = E\{x_{h+i}\}\, E\{y_{n+1-i}\}$$

and the terms γ_i (i=1, ...,n_{eq}) are independent, we can write:

$$\sum_{A \in \Theta} \gamma_i P(A) = \frac{1}{4} \\ \sum_{A \in \Theta} \gamma_i \gamma_j P(A) = \frac{1}{16} \\ \sum_{A \in \Theta} \gamma_i \gamma_j \gamma_m P(A) = \frac{1}{64} \tag{A.16}$$

Substituting (A.16) in the equation in eq. (A.15) one obtains:

$$\varepsilon^2_{\text{intrinsic}} = \mathbf{lsb}^2 \cdot 2^{-2h} \left[\frac{1}{24} 2^{-n_{eq}} - \frac{7}{324} 2^{-2n_{eq}} + \right. \\ \left. + \left(\frac{13}{864} - \frac{1}{48} 2^{-n_{eq}} \right) \cdot n_{eq} - \frac{13}{648} \right] \tag{A.17}$$

Appendix B

Calculation of Linear function

In order to solve the system (2.62) we can firstly compute the mean μ_{erasing} obtained when the linear compensation function (2.60) is employed. By substituting (2.60) and (2.54) in (2.29) we have:

$$\mu_{\text{lin}} = \sum_{A \in \Theta} \left(-H - \sum_{i=1}^{n_{eq}} \alpha_i \gamma_i - \sum_{i=1}^{Neq} \sum_{\substack{j=1 \\ j \neq i}}^{n_{eq}} \frac{f_{i,j}}{2} \gamma_i \gamma_j \right) \cdot P(A) \qquad \text{(B.1)}$$

where we have defined:

$$H = K - K_l; \quad \alpha_i = f_i - l_i \qquad \text{(B.2)}$$

Inverting the external sum over the set Θ with the internal sums on indexes i, j and using (A.16), the equation (B.1) can be simplified as follows:

$$\mu_{\text{lin}} = -H - \frac{1}{4} \sum_{i=1}^{n_{eq}} \alpha_i - \frac{1}{16} \sum_{i=1}^{n_{eq}} \sum_{\substack{j=1 \\ j \neq i}}^{n_{eq}} \frac{f_{i,j}}{2} \qquad \text{(B.3)}$$

We can now compute σ^2_{lin}. By substituting (B.3), (2.60) and (2.54) in (2.61)

we have:

$$\sigma^2_{\text{lin}} = \sum_{A\in\Theta}\left(\sum_{i=1}^{Neq}\alpha_i\left(\gamma_i - \frac{1}{4}\right) + \right.$$
$$\left. + \sum_{i=1}^{Neq}\sum_{\substack{j=1\\ j\neq i}}^{Neq}\frac{f_{i,j}}{2}\left(\gamma_i\gamma_j - \frac{1}{16}\right)\right) \tag{B.4}$$

Note that σ^2_{lin} does not depend on H, as observed in Sec. 2.4. In order to find the optimal l_i values we have to solve the system (2.62). By substituting (B.4) in the system (2.62) one obtains:

$$\sum_{A\in\Theta}\gamma_m\cdot\left(\sum_{i=1}^{Neq}\alpha_i\left(\gamma_i - \frac{1}{4}\right) + \right.$$
$$\left.\sum_{i=1}^{Neq}\sum_{\substack{j=1\\ j\neq i}}^{Neq}\frac{f_{i,j}}{2}\left(\gamma_i\gamma_j - \frac{1}{16}\right)\right)\cdot P(A) = 0 \quad m = 1,...,Neq \tag{B.5}$$

This system can be again simplified with the help of (A.16). With simple algebra the system simplifies directly in diagonal form:

$$\begin{cases} \alpha_m = -\dfrac{1}{2}\displaystyle\sum_{\substack{i=1\\ i\neq m}}^{n_{eq}}\frac{f_{i,m}}{2} & m = 1,...,n_{eq} \end{cases} \tag{B.6}$$

From (B.6) the result (2.63) is easily obtained.

As discussed in Sec. 2.4, K_l (that is H) can be fixed by imposing $\mu_{\text{lin}} = 0$. By using the μ_{lin} expression (B.3), we have:

$$-H - \frac{1}{4}\sum_{i=1}^{n_{eq}}\alpha_i - \frac{1}{16}\sum_{i=1}^{n_{eq}}\sum_{\substack{j=1\\ j\neq i}}^{n_{eq}}\frac{f_{i,j}}{2} = 0 \tag{B.7}$$

By substituting (B.6) in (B.7) and solving for H we found:

$$H = \frac{1}{16} \sum_{i=1}^{n_{eq}} \sum_{\substack{j=1 \\ j \neq i}}^{n_{eq}} \frac{f_{i,j}}{2} \tag{B.8}$$

From this equation the result (2.65) is easily verified.

Bibliography

[1] C. Baught and B. Wooley, "A two's complement parallel array multiplication algorithm," *IEEE Transactions on Computers*, vol. C-23, no. 12, pp. 1045–1047, Dec. 1974.

[2] C. Wallace, "A suggestion for fast multiplier," *IEEE Transaction on Electronic Computers*, vol. EC-13, no. 1, pp. 14–17, Feb. 1964.

[3] L. Dadda, "Some schemes for parallel multipliers," *Alta Frequenza*, vol. 34, pp. 349–356, Jan. 1965.

[4] K. Bickerstaff, M. Schulte, and J. E.E. Swartzlander, "Parallel reduced area multipliers," *Journal of VLSI Signal Processing Systems*, vol. 9, no. 3, pp. 181–191, Apr. 1995.

[5] V. Oklobdzija, D. Villeger, and S. Liu, "A method for speed optimized partial product reduction and generation of fast parallel multipliers using an algorithmic approach," *IEEE Transactions on Computers*, vol. 45, no. 3, pp. 294–309, Mar. 1996.

[6] P. Stelling, C. Martel, V. Oklobdzija, and R. Ravi, "Optimal circuits for parallel multipliers," *IEEE Transaction on Computers*, vol. 47, no. 3, pp. 273–285, Mar. 1998.

[7] J. Um and T. Kim, "Optimal bit-level arithmetic optimization for high-speed circuits," *Electronics Letters*, vol. 36, no. 5, pp. 405–406, Mar. 2000.

[8] S. Kidambi, F. El-Guibaly, and A. Antonious, "Area-efficient multipliers for digital signal processing applications," *IEEE Transaction on Circuits and Systems II: Analog and digital signal processing*, vol. 43, no. 2, pp. 90–95, Feb. 1996.

[9] Y. Lim, "Single-precision multiplier with reduced circuit complexity for signal processing applications," *IEEE Transactions on Computers*, vol. 41, no. 10, pp. 1333–1336, Oct. 1992.

[10] E. King and J. E.E. Swartzlander, "Data dependent truncated scheme for parallel multiplication," in *Proc. of 31th Asilomar Conf. on signals, circuits and systems*, 1998, pp. 1178–1182.

[11] M. Schulte and J. E.E. Swartzlander, "Truncated multiplication with correction constant [for DSP]," in *VLSI Signal Processing, VI, 1993., [Workshop on]*, Veldhoven, Netherlands, Oct. 1993, pp. 388–396.

[12] J. Stine and O. Duverne, "Variations on truncated multiplication," in *Proceedings of the Euromicro Symposium on Digital Systems Design, DSD '03*, Sep. 2003, pp. 112– 119.

[13] J. Jou, S. Kuang, and R. Chen, "Design of low-error fixed-width multipliers for DSP applications," *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, vol. 46, no. 6, pp. 836–842, Jun. 1999.

[14] F. Curticapean and J. Niittylahti, "A hardware efficient direct digital frequency synthesizer," in *The 8th IEEE International Conference on Electronics, Circuits and Systems, 2001. ICECS 2001*, vol. 1, Malta, May 2001, pp. 51–54.

[15] L. Van, S. Wang, and W. Feng, "Design of the lower error fixed-width multiplier and its application," *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, vol. 47, no. 10, pp. 1112–1118, Oct. 2000.

[16] L. Van and C. Yang, "Generalized low-error area-efficient fixed-width multipliers," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 52, no. 8, pp. 1608–1619, Aug. 2005.

[17] Y. Liao, H. Chang, and C. Liu, "Carry estimation for two's complement fixed-width multipliers," in *IEEE Workshop on Signal Processing Systems Design and Implementation, 2006. SIPS '06.*, Banff, Alta., Oct. 2006, pp. 345–350.

[18] S. Kuang and J. Wang, "Low-error configurable truncated multipliers for multiply-accumulate applications," *Electronics Letters*, vol. 42, no. 16, pp. 904–905, Aug. 2006.

[19] R. Michard, A. Tisserand, and N. Charvillon, "Carry prediction and selection for truncated multiplication," in *IEEE Workshop on Signal Processing Systems Design and Implementation, 2006. SIPS '06.*, Banff, Alta., Oct. 2006, pp. 339–344.

[20] H. Park and J. E.E. Swartzlander, "Truncated multiplication with symmetric correction," in *Fortieth Asilomar Conference on Signals, Systems and Computers, 2006. ACSSC '06.*, Pacific Grove, CA, Nov. 2006, pp. 931–934.

[21] A. Strollo, N. Petra, and D. D. Caro, "Dual-tree error compensation for high performance fixed-width multipliers," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 52, no. 8, pp. 501–507, Aug. 2005.

[22] B. Parhami, *Computer Arithmetic: Algorithms and Hardware Designs*. Oxford University Press, 1999.

[23] J. E.E. Swartzlander, "Truncated multiplication with approximate rounding," in *Conference Record of the Thirty-Third Asilomar Conference on Signals, Systems, and Computers, 1999.*, vol. 2, Pacific Grove, CA, USA, Oct. 1999, pp. 1480–1483.

[24] C. Piguet, *Low-power electronics design*. CRC Press, 2005.

[25] N. Petra, D. D. Caro, and A. Strollo, "Minimum mean square error signed and unsigned inputs fixed-width multipliers," in *Proceedings of IEEE European Conference on Circuits Theory and Design (ECCTD07)*, Sevilla, Spain, Aug. 2007, pp. 464–467.

[26] J. Pihl and E. Aas, "A multiplier and square generator for high performance dsp applications," in *Proceedings of the 39th Midwest IEEE Symposium on Circuit and Systems*, vol. 1, Ames, IA, USA, Aug. 1996, pp. 109 – 112.

[27] R. Kologotla, W. Griescbach, and H. Scrinivas, "Vlsi implementation of a 350mhz 0.35 um 8 bit merged squarer," *Electronics Letters*, vol. 34, pp. 47 – 48, Jan. 1998.

[28] K. Wires, M. Schulte, L. Marquette, and P. Balzola, "Combined unsigned and two's complement squarers," in *33rd Asilomar Conference on Signals, Systems and Computers*, vol. 2, Pacific Grove, CA, USA, Oct. 1999, pp. 1215 – 1219.

[29] A. Strollo, E. Napoli, and D. D. Caro, "New design of squarer circuits using booth encoding and folding techniques," in *The 8th IEEE International Conference on Electronics, Circuits and Systems, ICECS 2001*, vol. 1, Sep. 2001, pp. 193 – 196.

[30] A. Strollo and D. D. Caro, "Booth folding encoding for high performance squarer circuits," *IEEE Transaction on Circuits and Systems II: Analog and Digital Signal Processing.*, vol. 50, no. 5, pp. 250 – 254, May 2003.

[31] T. Chen, "A binary multiplication scheme based on squaring," *IEEE Transactions on Circuits and Systems I*, vol. C, no. 20, pp. 678 – 680, 1971.

[32] R. Strandberg, L. Bustamante, V. Oklobdzija, M. Soderstrand, and J. LeDuc, "Efficient realizations of squaring circuit and reciprocal used in adaptive sample rate notch filters," *IEEE Journal of VLSI Signal Processing*, vol. 14, no. Dec., pp. 303 – 309, 1996.

[33] E. W. III and M. Schulte, "Efficient function approximation using truncated multipliers and squarers," in *Proceedings of the 17th IEEE Symposium on Computer Arithmetic*, Jun. 2005, pp. 232 – 239.

[34] K. Cho, Y. Kim, and J. Chung, "Power and area efficient squarer design," in *Fortieth Asilomar Conference on Signals, Systems and Computers, 2006. ACSSC '06*, Pacific Grove, CA, Nov. 2006, pp. 1721–1725.

[35] A. Oppenheim and R. Schafer, *Discrete-Time Signal Processing*. Prentice Hall, 1999.

[36] S. Smith, *The scientist and Engineers Guide to Digital Signal PRocessing*. California Technical Publishing, 1999.

[37] R. Lyons, *Understanding Digital Signal Processing*. Prentice Hall Professional Technical Reference, 2004.

[38] I. Simon, N. Barsan, and U. Weimar, "Micromachined metal oxide gas sensors: Opportunities to improve sensor performance," *Sensors and Actuators B: Chemical*, vol. 73, no. 1, pp. 1–26, Feb. 2001.

[39] S. Santra, S. Ali, P. Guha, P. Hiralal, H. E. Unalan, S. Dalal, J. Covington, J. Gardner, W. Milnea, and F. Udrea, "Cmos alcohol sensor employing zno nanowire sensing films," in *Proceedings of the 13th International*

Symposium on Olfaction and Electronic Nose. AIP Conference Proceedings, vol. 1137, Brescia,Italy, Oct. 2009, pp. 119–122.

[40] J. Gardner, V. Varadan, and O. Awadelkarim, *Microsensors MEMS and Smart Devices*. John Wiley & Sons, 2001.

[41] P. Guha, S. Ali, C. Lee, F. Udrea, W. Milne, T. Iwaki, J. Covington, and J. Gardner, "Novel design and characterisation of soi cmos microhotplates for high temperature gas sensors," *Sensors and Actuators B: Chemical*, vol. 127, no. 1, pp. 260–266, 2007.

[42] J. Suehle, R. Cavicchi, M. Gaitan, and S. Semancik, "Tin oxide gas sensor fabricated using cmos micro-hotplates and in-situ processing," *IEEE Electron Device Letters*, vol. 14, no. 3, pp. 118–120, Mar. 1993.

[43] F. Udrea, J. Gardner, D. Setiadi, J. Covingtom, T. Dogaru, C. Lu, and W. Milne, "Design and simulations of soi cmos micro-hotplate gas sensors," *Sensors and Actuators B: Chemical*, vol. 78, pp. 180–190, Aug. 2001.

[44] M. Afridi, J. Suehle, M. Zaghloul, D. Berning, A. Hefner, R. Cavacchi, S. Semancik, C. Montgomery, and C. Taylor, "A monolithic cmos microhotplate-based gas sensor system," *IEEE Sensors Journal*, vol. 2, no. 6, pp. 644–655, Dec. 2002.

[45] S. Ali, F. Udrea, W. Milne, and J. Gardner, "Micromachined metal oxide gas sensors: Opportunities to improve sensor performance," *Journal of Microelectromechanical system*, vol. 17, no. 6, pp. 1408–1414, Dec. 2008.

[46] M. Graf, D. Barrettino, H. Baltes, and A. Hierlemann, *CMOS Hotplate Chemical Microsensors*. Springer, 2005.

[47] S. Santra, P. Guha, S. Ali, I. Haneef, F. Udrea, and J. Gardner, "Soi diode temperature sensor operated at ultra high temperature - a critical analysis," in *Proc. of 13th IEEE Sensors Conference*, Italy, Oct. 2008, pp. 78–81.