

UNIVERSIDAD POLITÉCNICA DE MADRID

**ESCUELA TÉCNICA SUPERIOR
DE INGENIEROS DE TELECOMUNICACIÓN**



**COMBINED WORD-LENGTH ALLOCATION AND
HIGH-LEVEL SYNTHESIS OF DIGITAL SIGNAL
PROCESSING CIRCUITS**

TESIS DOCTORAL

GABRIEL CAFFARENA FERNÁNDEZ
INGENIERO DE TELECOMUNICACIÓN
2008

DEPARTAMENTO DE INGENIERÍA ELECTRÓNICA
ESCUELA TÉCNICA SUPERIOR DE INGENIEROS DE
TELECOMUNICACIÓN



TESIS DOCTORAL

COMBINED WORD-LENGTH ALLOCATION AND
HIGH-LEVEL SYNTHESIS OF DIGITAL SIGNAL
PROCESSING CIRCUITS

Autor:

Gabriel Caffarena Fernández
Ingeniero de Telecomunicación

Directores:

Octavio Nieto-Taladriz García
Doctor Ingeniero de Telecomunicación

Carlos Carreras Vaquer
Doctor Ingeniero de Telecomunicación

2008

© 2008 Gabriel Caffarena Fernández

All rights reserved. No part of this publication may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopy, recording, or any information storage and retrieval system, without the permission from the author.

To Susana

CONTENTS

Contents	i
Abstract	v
Acknowledgments	vii
List of Figures	ix
List of Tables	xi
List of Acronyms	xiii
List of Notation	xv
Resumen	xix
1 Introduction	1
1.1 Motivation	2
1.2 Objectives	5
1.3 Book organization	6

1.4	Contributions of the thesis	6
1.5	Publications	8
2	State of the art	15
2.1	Word-length allocation	16
2.2	High-Level synthesis	23
2.3	Combined word-length allocation and high-level synthesis	27
2.4	Field-programmable gate arrays	30
2.5	Conclusions	32
3	High-level synthesis of DSP algorithms using FPGAs	35
3.1	High-level synthesis	36
3.2	Cost estimation	41
3.2.1	Resource usage metric	41
3.2.2	Resource modeling	46
3.3	Optimization procedure	53
3.4	Results	59
3.4.1	Uniform word-length vs. multiple word-length synthesis: Homogeneous architectures	62
3.4.2	Uniform word-length vs. multiple word-length synthesis: Heterogeneous architectures	65
3.4.3	MWL synthesis: Heterogeneous vs. Homogeneous	69
3.4.4	Effect of registers and multiplexers	72
3.5	Conclusions	74
4	Word-length allocation	77
4.1	Word-length Allocation	78
4.2	Scaling and error estimation	86

4.2.1	Affine arithmetic	86
4.2.2	Scaling	87
4.2.3	Error estimation	88
4.3	Word-Length selection	96
4.4	Results	100
4.4.1	Error estimation results	101
4.4.2	Word-length selection techniques comparison	105
4.5	Conclusions	114
5	Combined approach	117
5.1	Optimal approach	118
5.1.1	Problem definition	119
5.1.2	MILP formulation	122
5.1.3	Results and conclusions	127
5.2	Heuristic approach	130
5.2.1	Optimization procedure	131
5.2.2	Results and conclusions	135
5.3	Conclusions	147
6	Conclusions	151
6.1	Conclusions	151
6.2	Future research lines	158
	Bibliography	161

ABSTRACT

This work is focused on the synthesis of Digital Signal Processing (DSP) circuits using specific hardware architectures. Due to its complexity, the design process has been subdivided into separate tasks, thus hindering the global optimization of the resulting systems. The author proposes the study of the combination of two major design tasks, Word-Length Allocation (WLA) and High-Level Synthesis (HLS), aiming at the optimization of DSP implementations using modern Field Programmable Gate Array devices (FPGAs).

A multiple word-length approach (MWL) is adopted since it leads to highly optimized implementations. MWL implies the customization of the word-lengths of the signals of an algorithm. This complicates the design, since the number possible assignments between algorithm operations and hardware resources becomes very high. Moreover, this work also considers the use of heterogeneous FPGAs where there are several types of resources: configurable logic-based blocks (LUT-based) and specialized embedded resources. All these issues are addressed in this work and several automatic design techniques are proposed.

The contributions of the Thesis cover the fields of WLA, HLS using FPGAs, and the combined application of WLA and HLS for implementation in FPGAs.

A thorough approach of HLS has been implemented which considers a complete datap-

ath composed of functional units (FUs), registers and multiplexers, as well as heterogeneous FPGA resources (LUT-based and embedded resources). The approach makes use of a resource library that accounts for MWL effects within the set of resources, thus producing highly optimized architectures. This library includes both LUT-based and embedded FPGA resources, which further increase the power of the HLS task. Another important contribution is the introduction of resource usage metrics suitable for heterogeneous-architecture FPGAs.

A novel quantization error estimation based on affine arithmetic (AA) is presented, as well as its practical application to the automatic WLA of LTI and non-linear differentiable DSP systems. The error estimation is based on performing a pre-processing of the algorithm, which produces an expression of the quantization error at the system output. Therefore, the error can be easily computed leading to fast and accurate WLA optimizations.

The analysis of the impact of different optimization techniques during WLA on HLS results is also presented. The variance in the obtained results corroborates the fact that it is worth using a single architecture model during WLA and HLS, and this is only possible by means of combining these tasks.

The actual combination of WLA and HLS has been firstly performed by using a Mixed Integer Linear Programming (MILP) approach. The results prove the validity of the approach and also provide with insights into the combination of the two tasks that are used to generate heuristic synthesis algorithms.

Finally, the global contribution of this thesis is an HLS heuristic algorithm able to perform the combined WLA and HLS of DSP systems for both homogeneous and heterogeneous FPGA architectures. Up to 20% of resource usage reductions are reported, which proves the importance of such a combined approach, providing electronic designers with a design framework that enables highly improved DSP custom hardware implementations.

ACKNOWLEDGMENTS

Here, in these lines I take the opportunity to thank all the people that have given me support during this research period.

First of all I would like to thank both my advisors, Dr. Nieto-Taladriz and Dr. Carreras, for the opportunity given and the effort they have put into this. I specially thank Dr. Carreras for his invaluable help and support. This gratitude must be extended to the members of the Departamento de Ingeniería Electrónica, Universidad Politécnica de Madrid.

I also would like to thank the researchers that supervised my work during research visits around Europe. Veerle Derudder, André Bourdoux and Jan-Willem Weijers were most kind and helpful during my visit to the Inter-University Microelectronics Center (Leuven, Belgium). At Imperial College London, I was under the attentive supervision of Peter Y.K. Cheung and George Constantinides.

I must also mention my MSc supervisor Pelegrín Camacho from the Departamento de Tecnología Electrónica, Universidad de Málaga. He was as wise as to suggest me a master thesis on reconfigurable computing.

I thank my colleagues Juan Antonio López, Gerardo Leyva, Angel Fernández and Jose Ayala for the technical support. Thank you for your precious time. I am also lucky enough

to have them as good friends.

I have very good memories of the early days when lunch turned into an interesting (and funny) debate everyday. The aforementioned as well as Marisa López, Jose Manuel Moya, Antonio Lomeña and Juan Marcos Díez completed the *old gang*. I had also many teckie and not-so-teckie talks with Miguel Angel Sánchez. Now, new blood has arrived: I wish a bright research period to Ruzica Jevtic, Zorana Bankovic, Vule Pejovic, Javier Gónzalez, Pablo Ituero and Pedro Echeverría, who suffer my compulsive talking during lunch.

Marisa and Jose, many thanks for the \LaTeX tips.

Juan Antonio deserves a special mention, since he is the one to blame for this research. He has proved to be an excellent colleague and a better friend. Thanks.

My family. I could not be more fortunate. My parents, my brother, my sister, my parents in law, my bothers and sisters in law, my nephews. You have provided companionship and love despite the distance.

My family. I thank my parents for the love and support and for the education given in all senses.

My family. Susana, wife, mother, translator, you name it. Infinite patience, infinite love. Every single *bit* of this research is dedicated to you. Thank you.

My family. Gabriel, my son. I love you. Don't worry, you won't have to read this book.

Finally, a very special memory to Jose David Romeral.

LIST OF FIGURES

3.1	Conversion from G_{SFG} (a) into G_{SEQ} (b)	37
3.2	Fixed-point format	38
3.3	Datapath architecture: FUs, registers and multiplexers	38
3.4	Normalized resource usage using ∞ -norm	42
3.5	Normalized resource usage using 1-norm	43
3.6	Normalized resource usage using $+$ -norm	45
3.7	Computation of addition core word-length	48
3.8	Different signal alignment approaches: a) arbitrary alignment, b) LSB alignment, c) MSB alignment	51
3.9	UWL vs. MWL: homogeneous implementations (I)	62
3.10	UWL vs. MWL: homogeneous implementations (II)	63
3.11	UWL vs. MWL: heterogeneous implementations (I)	66
3.12	UWL vs. MWL: heterogeneous implementations (II)	67
3.13	MWL synthesis: homogeneous vs. heterogeneous (I)	70
3.14	MWL synthesis: homogeneous vs. heterogeneous (II)	71

4.1	Word-length allocation diagram.	79
4.2	Signals' fixed-point information	81
4.3	Quantization of forks: a) naive approach; b) accurate cascade model	84
4.4	Resource-sharing HLS: Homogeneous architectures	109
4.5	Resource-sharing HLS: Heterogeneous architectures	111
5.1	<i>SEQ</i> vs. <i>COMB</i> : homogeneous implementations (I)	137
5.2	<i>SEQ</i> vs. <i>COMB</i> : homogeneous implementations (II)	138
5.3	<i>SEQ</i> vs. <i>COMB</i> : homogeneous implementations (III)	139
5.4	<i>SEQ</i> vs. <i>COMB</i> : heterogeneous implementations (I)	142
5.5	<i>SEQ</i> vs. <i>COMB</i> : heterogeneous implementations (II)	143
5.6	<i>SEQ</i> vs. <i>COMB</i> : heterogeneous implementations (III)	144

LIST OF TABLES

2.1	Summary of WLA approaches.	18
2.2	Summary of HLS approaches.	24
2.3	Summary of combined WLA and HLS approaches.	27
3.1	Area and delay parameters of FUs and registers.	50
3.2	Characteristics of benchmarks.	61
3.3	UWL vs. MWL for homogeneous architectures.	65
3.4	UWL vs. MWL for heterogeneous architectures.	68
3.5	MWL synthesis: homogeneous vs. heterogeneous architectures.	72
3.6	Complete vs. simplified cost estimation: Area improvement (%).	73
3.7	Complete vs. simplified cost estimation: Optimization time.	73
4.1	Error estimation results for 5th-order LMS filter (LMS_5).	102
4.2	Accuracy of the estimation method.	104
4.3	Computational performance of noise estimation method.	105
4.4	UWL vs. $GRAD$, $GRAD_2$, SA : direct homogeneous implementations. . .	106
4.5	UWL vs. $GRAD$, $GRAD_2$, SA : direct heterogeneous implementations. .	108

4.6	<i>GRAD</i> vs. <i>GRAD</i> ₂ , <i>SA</i> : resource-sharing homogeneous architectures. . .	110
4.7	<i>GRAD</i> vs. <i>GRAD</i> ₂ , <i>SA</i> : resource-sharing heterogeneous architectures. . .	112
4.8	Optimization times for <i>GRAD</i> , <i>GRAD</i> ₂ and <i>SA</i>	113
5.1	Area reduction (%) obtained by the optimal combined approach	127
5.2	Detailed word-lengths for <i>FIR</i> ₃	128
5.3	Characteristics of benchmarks.	136
5.4	<i>SEQ</i> vs. <i>COMB</i> : homogeneous implementations	140
5.5	<i>SEQ</i> vs. <i>COMB</i> : heterogeneous implementations	145
5.6	Optimization times: <i>SEQ</i> vs. <i>COMB</i>	146

LIST OF ACRONYMS

AA	Affine Arithmetic.
ASA	Adaptive Simulated Annealing.
ASIC	Application Specific Integrated Circuit.
CAD	Computer-Aided Design.
CLB	Configurable Logic Block.
COMB	Combined application of WLA and HLS.
DCT	Discrete Cosine Transform.
DFG	Data Flow Graph.
DSP	Digital Signal Processing.
EDA	Electronic Design Automation.
EXS	Exhaustive Search.
FPGA	Field Programmable Gate Array.
FIR	Finite Impulse Response.
FU	Functional Unit.
HOM	Homogeneous-architecture approach.
HET	Heterogeneous-architecture approach.

HLS	High-Level Synthesis.
IA	Interval Arithmetic.
IIR	Infinite Impulse Response.
IOB	Input/Output Block.
LE	Logic Element.
LMS	Least Mean Squares.
LSB	Least Significant Bit.
LUT	Look-Up Table.
MILP	Mixed Integer Linear Programming.
MSB	Most Significant Bit.
MSE	Mean Square Error.
MUX	Multiplexer.
MWL	Multiple Word-Length.
RTL	Register Transfer Logic.
SA	Simulated Annealing.
SEQ	Sequential application of WLA and HLS.
SFG	Signal Flow Graph.
SNR	Signal to Noise Ratio.
SQNR	Signal to Quantization Noise Ratio.
UWL	Uniform Word-Length.
WLA	Word-Length Allocation.

LIST OF NOTATION

$\ \cdot\ _p$	p-norm of enclosed vector.
$\ \cdot\ _{+p}$	+p-norm of enclosed vector.
$a \vee b$	The disjunction of terms a and b .
$a \wedge b$	The conjunction of terms a and b .
$\neg a$	Logical NOT.
$A \setminus B$	Set subtraction between sets A and B .
$ A $	Cardinality number of set A .
$out(v)$	Number of output edges of node v .
$in(v)$	Number of input edges of node v .
\hat{A}	Normalized FPGA Area vector.
\hat{A}_{LUT}	Normalized number of LUT-based resources.
\hat{A}_{EMB}	Normalized number of embedded resources.
$A_{i,n}$	Amplitude of error $\epsilon_{i,n}$ at the output of the algorithm.
\hat{x}	Affine form.

$\beta(o)$	Resource binding function. Outputs the resource/instance couple $\{r, i\}$ for operation o .
ϵ_i	i -th error term of affine form.
$\epsilon_{i,n}$	Error term associated to signal s_i and generated at time-step n .
$\gamma(d)$	Register binding function. Outputs the register/instance couple $\{r, i\}$ for variable d .
$\varphi(o)$	Scheduling function. Outputs the starting time of operation o .
λ	Latency of algorithm.
$\lambda_r, \lambda(r)$	Latency of resource r .
C	Set of compatibility edges (G_{COMP}).
E	Set of data dependency edges (G_{SEQ}).
E_{max}	Output noise constraint at the output of the algorithm.
D	Set of algorithm variable.
$D(r)$	Delay in nanoseconds of resource r .
$g_i[n]$	Impulse response from signal s_i to output.
$G_i(Z)$	Transfer function from signal s_i to output.
G_{COMP}	Compatibility Graph.
G_{SEQ}	Sequence Graph.
G_{SFG}	Signal Flow Graph.
$h[n]$	Impulse response.
$H(Z)$	Transfer function.
K	Constant used to compute +-norm.
$L_2(H(Z))$	L_2 norm of transfer function $H(Z)$.

n_i	Word-length of signal s_i .
n_i^{pre}	Pre-quantization word-length of signal s_i .
$n_{core,v}$	Core word-length of addition v .
O	Set of algorithm operations (G_{COMP}).
p_i	Scaling of signal s_i .
p_i^{pre}	Pre-quantization scaling of signal s_i .
S	Set of signals (G_{SFG}).
s_i	Signal with index i .
R	Set of resources.
R_T	Set of FPGA resources of type T .
$\#r_T$	Number of FPGA resources of type T used for a design implementation.
T_{ck}	Clock period.
V	Set of algorithm operations (G_{SFG}).

MIXED INTEGER LINEAR PROGRAMMING (CHAPTER 5.1)

A_r	Area of adder r .
\hat{n}_i	Upper bound of the word-length of signal s_i .
\bar{n}	Binary variable use to linearize the expression 2^{-2n} .
$x_{o,t,r}$	Binary scheduling and resource binding variable for operation o , time step t and resource r
$T(o)$	Execution time interval for operation(o).
T	Total execution time interval for algorithm.
$\alpha(i)$	Signal index of $i - th$ output of a fork.

E_v	Quantization error generated by fork v .
$\delta_{a_1}, \delta_{a_2}$	Binary decision variables used in the computation of the area of adder a .
β_v	Word-length of safe adder implementation of addition v
m_v	Number of unnecessary bits at the output of addition v
ϵ, η	Binary variables used in the computation of the error of forks.

RESUMEN

El procesamiento digital de la señal (DSP: Digital Signal Processing) desempeña un papel fundamental en el desarrollo de sistemas multimedia y de comunicaciones. El continuo crecimiento de la demanda por parte de los usuarios de aplicaciones cada vez más complejas sólo puede ser contrarrestado mediante el desarrollo simultáneo de las técnicas DSP y de las técnicas microelectrónicas. Por ejemplo, los sistemas de comunicaciones inalámbricos modernos requieren el procesamiento de cientos de canales de datos de alta velocidad, y este hecho complica el diseño, puesto que tanto los algoritmos DSP como su implementación mediante sistemas electrónicos están siendo llevados al límite de las posibilidades actuales.

La gran complejidad de estos sistemas digitales hace necesaria la introducción de herramientas potentes para el diseño asistido por ordenador (CAD: Computer-Aided Design) que permitan llevar a cabo su implementación física. Cualquier aportación en el estado del arte de los algoritmos utilizados por las herramientas CAD es siempre bienvenida, puesto que puede dar lugar finalmente a reducciones en los costes de diseño alcanzables hasta la fecha. De hecho, las compañías dedicadas a la elaboración de software para el Diseño Electrónico Automático (EDA: Electronic Design Automation) están continuamente evolucionando para así satisfacer las exigencias de los usuarios que demandan aplicaciones más complejas.

Ello conlleva un continuo intento de explotar al máximo las posibilidades de los avances en el mundo de la microelectrónica. Es en esta línea, el desarrollo de una generación de herramientas de diseño especialmente enfocado a tecnologías reconfigurables tipo FPGA (Field-Programmable Gate Arrays), en la que la presente Tesis está enmarcada.

El ciclo de diseño de sistemas electrónicos se divide en varias tareas, lo cual permite al diseñador recorrer el camino que existe desde las especificaciones del diseño hasta su implementación. El análisis de la posible combinación de dos tareas de diseño sobradamente conocidas, la Asignación de Anchos de Palabras (WLA: Word-Length Allocation) y la Síntesis de Alto Nivel (HLS: High-Level Synthesis), aplicadas a la producción de arquitecturas *hardware* específicas para DSP, es el tema principal de este trabajo de investigación, teniendo como objetivo principal la mejora de las técnicas actuales para el diseño de sistemas DSP sobre FPGAs.

MOTIVACIÓN

La Asignación de Anchos de Palabras, también denominada *cuantificación*, es la traducción de una descripción inicial de un algoritmo utilizando precisión infinita a una descripción con precisión finita. La aritmética en coma fija (*fixed-point* en inglés) es comúnmente utilizada para la implementación de circuitos DSP, ya que ha demostrado que permite conseguir implementaciones de bajo coste, bajo consumo y alta velocidad si se la compara con implementaciones basadas en el uso de aritmética en coma flotante (*Floating-point*). Por lo tanto, en este trabajo el término cuantificación se entiende como el proceso mediante el cual se realiza la traducción de una descripción de un algoritmo con precisión infinita (existe el convenio de utilizar el formato de coma flotante IEEE 754-1984) a una descripción basada en coma fija, dada una restricción de error máximo a las salidas del algoritmo.

Es necesario realizar la selección de los anchos de palabra de las diferentes señales que componen la implementación hardware de forma inteligente, para así conseguir minimizar los costes de diseño. Tradicionalmente, se ha usado el enfoque denominado de Anchos de Palabra Uniformes (UWL: Uniform Word-Length), que se heredó del diseño basado en microprocesadores. El enfoque UWL permitía realizar la cuantificación de sistemas de forma muy rápida, pero desperdiciaba recursos hardware. En las últimas dos décadas, este enfoque ha ido cayendo en desuso debido a la aparición de un enfoque basado en el uso de Anchos de Palabra Múltiples (MWL: Multiple Word-Length). El enfoque MWL se basa en seleccionar de forma individual los anchos de palabra de cada señal obteniéndose así grandes reducciones de coste. Sin embargo, el paradigma MWL implica una mayor complejidad en las técnicas de WLA y HLS.

EL objetivo principal de la HLS es explorar el espacio de diseño para encontrar la mejor arquitectura hardware que cumpla con las restricciones y requerimientos de potencia, área y velocidad de procesamiento. HLS está compuesta principalmente por las técnicas de planificación temporal (*scheduling*), selección de recursos (*resource allocation*) y asignación de recursos (*resource binding*). HLS convierte una especificación inicial del algoritmo (por ejemplo un Grafo de Flujo de Datos (DFG: Data Flow Graph)) en una descripción a Nivel de Transferencia de Registros (RTL: Register Transfer Level), cumpliendo con las restricciones de diseño.

Las tareas de WLA y HLS se han realizado tradicionalmente de forma separada para reducir su complejidad computacional. Primero se ejecuta WLA para obtener los anchos de palabras de las variables del algoritmo. Después de esto, HLS analiza la descripción del algoritmo cuantificado y genera la arquitectura final que será implementada en el dispositivo hardware. La desventaja principal de esta metodología es que no tiene en cuenta las interdependencias existentes entre las dos técnicas. Los anchos de palabra se seleccionan normalmente asignándoles inicialmente un valor elevado y reduciendo su valor gradualmente

hasta que se llega al límite en el que se cumplen las restricciones de error. En esta etapa es habitual no tener en cuenta detalles arquitecturales, y esto reduce el problema significativamente. Por ejemplo, el objetivo de la WLA puede ser minimizar el número total de bits correspondiente a la suma de los anchos de palabra de cada variable, ya que esto a grandes rasgos reduce el área de los recursos aritméticos, la cantidad de área destinada al cableado, etc. Sin embargo, este enfoque no garantiza que los resultados sean óptimos. Existen otras aproximaciones al problema que incluyen un modelo de la arquitectura, el cual se usa para guiar el proceso de optimización en el que se seleccionan los anchos de palabra. Sin embargo, este modelo suele basarse en una implementación totalmente paralela del algoritmo, sin considerar que posteriormente, durante la HLS, pueden reutilizarse recursos. Como consecuencia, no está garantizado que el coste final del sistema tras la síntesis (HLS) sea mínimo, puesto que los modelos arquitecturales utilizados durante WLA y durante HLS difieren.

La aplicación conjunta de ambas técnicas permitiría el uso de un único modelo arquitectural durante todo el proceso de diseño, lo cual conllevaría la obtención de implementaciones altamente optimizadas, debido a que es posible llevar a cabo una exploración del espacio de diseño más extensa. El precio a pagar es la generación de nuevas técnicas que permiten abordar el problema combinado, y debido al hecho de que tanto WLA como HLS son ya de por sí tareas muy complejas, la combinación de ambas se espera que posea una complejidad aún mayor. Este último detalle puede ser fácilmente solucionado en parte gracias a los continuos avances en la potencia de cálculo de los ordenadores, lo cual permite a científicos e ingenieros enfrentarse cada día a problemas cada vez más complejos. Aún así, todavía es necesario desarrollar nuevas técnicas que aborden el problema combinado de forma eficiente. A día de hoy, el enfoque combinado ha sido estudiado por unos pocos grupos de investigación, y sólo de forma superficial, y esto hace que existan muchas líneas de investigación todavía sin desarrollar.

Otro factor importante es la correcta adaptación a la tecnología de destino que estamos considerando. Los dispositivos programables, y en concreto los dispositivos FPGA, han ganado una gran relevancia en los últimos años. Sus prestaciones los sitúan en el límite entre los Circuitos Integrados de Aplicación Específica (ASIC: Application Specific Integrated Circuits) y los sistemas basados en microprocesador. Básicamente, estos dispositivos permiten la implementación de arquitecturas dedicadas que pueden ser reconfiguradas. Las posibilidades de reconfiguración que ofrecen hacen que sean ideales como plataformas de prototipado de sistemas complejos. Al mismo tiempo, los dispositivos más modernos presentan altas prestaciones en lo referente al nivel de integración y a la potencia de cálculo, lo cual hace que también resulten atractivos como productos finales. El mundo académico ha recibido estos dispositivos con los brazos abiertos por la posibilidad que ofrecen de utilizar indefinidas veces el mismo dispositivo para implementar diferentes sistemas, y también por la gran potencia de las herramientas de diseño existentes. Por estas razones se han seleccionado las FPGAs como la tecnología de implementación en este trabajo. Los últimos avances de los dispositivos reconfigurables, tales como la introducción de bloques empotrados especializados, ofrecen nuevas posibilidades para la implementación de algoritmos DSP de altas prestaciones. Se están realizando numerosos esfuerzos para encontrar nuevos modelos arquitecturales, así como algoritmos de diseño automáticos, que permitan sacarle el máximo provecho a la capacidad de las FPGAs. Por ejemplo, existen propuestas que sugieren el abandono del paradigma de las arquitecturas homogéneas (FPGAs basadas en LUTs (Look-up Tables)) y adoptar el paradigma de las arquitecturas heterogéneas (FPGAs con bloques basados en LUTs y bloques empotrados especializados). Este enfoque tan ambicioso es el que se sigue en nuestro trabajo.

OBJETIVOS

Los principales objetivos de esta Tesis son:

- Análisis detallado del estado del arte de las técnicas WLA y HLS, así como de la aplicación conjunta de éstas.
- Generación de un método completo de HLS que sea capaz de manejar:
 - Algoritmos DSP basados en descripciones MWL.
 - FUs, recursos de almacenamiento y lógica de multiplexación de forma simultánea.
 - Recursos de FPGAs heterogéneas (basados en LUTs y empotrados).
- Análisis comparativo de los resultados obtenidos por la HLS utilizando FPGAs con arquitecturas homogéneas y heterogéneas.
- Generación de un estimador del error de cuantificación que sea rápido y preciso para sistemas LTI y sistemas no lineales.
- Análisis del efecto del modelo arquitectural utilizado durante WLA en los resultados obtenidos tras la HLS.
- Análisis del efecto de la utilización de técnicas de optimización diferentes durante la WLA en los resultados obtenidos tras la HLS.
- Análisis óptimo del problema que combina WLA y HLS basado en técnicas de programación lineal.
- Generación de un algoritmo heurístico de síntesis que permita combinar WLA y HLS.
- Análisis de los resultados de la combinación de WLA y HLS para FPGAs con arquitecturas homogéneas con respecto a los obtenidos aplicando el enfoque tradicional secuencial.

- Análisis de los resultados de la combinación de WLA y HLS para FPGAs con arquitecturas heterogéneas.

CONTRIBUCIONES DE LA TESIS

Esta Tesis aborda la combinación de la Asignación de Anchos de Palabra y la Síntesis de Alto Nivel de circuitos para el Procesamiento Digital de la Señal. Estas dos técnicas de diseño están centradas en la optimización de circuitos DSP, de ahí el interés en su aplicación conjunta, ya que ésta permite alcanzar niveles de optimización superiores a los conseguidos por su aplicación secuencial. La WLA tiene como objetivo la optimización del coste (área, consumo y velocidad) mediante la selección de la precisión de las operaciones aritméticas involucradas en el procesamiento de la señal. La HLS de circuitos DSP tiene el mismo objetivo de optimización del coste, pero se centra en la elaboración de una arquitectura que minimice el desperdicio de recursos. Por una lado, la HLS depende mucho de la WLA, ya que el número de bits asignados a las señales determina el tamaño de las unidades funcionales (FU: Functional Unit), buses, etc; por otro lado, realizar una WLA óptima implicaría el conocimiento de la arquitectura final, y esto no es posible, puesto que dicha arquitectura es el resultado de la HLS. Por todo ello, resulta interesante combinar el análisis de la precisión matemática y la exploración de las posibles arquitecturas en una única tarea de diseño. Durante la persecución del objetivo principal de esta Tesis, la combinación de WLA y HLS, se han realizado aportaciones al estado del arte que no sólo incluyen aquellas relacionadas con el problema combinado, sino que también atañen a las tareas de WLA y HLS por separado.

La Síntesis de Alto Nivel orientada a sistemas MWL ha sido abordada teniendo como objetivo adicional su adaptación a las FPGAs modernas especializadas en el procesamiento DSP. El optimizador se ha basado en recocido simulado (SA: Simulated Annealing) debido a la flexibilidad que ofrece para optimizar problemas complejos y a sus prestaciones que permiten alcanzar resultados cuasi-óptimos. SA también es conocido por los altos tiempos de computación que se requieren para su convergencia. Sin embargo, el punto de vista adoptado en este trabajo de investigación es el de analizar las posibilidades de la HLS orientada a sistemas MWL (al igual que las posibilidades de la optimización combinada) y no se ha centrado en la eficiencia computacional, que es vista como una fase de investigación futura. El modelado del coste de los recursos (por ejemplo, área y latencia) ha sido un factor clave a la hora de orientar el algoritmo de HLS propuesto hacia sistemas MWL. A pesar de que existen modelos similares en la literatura, uno de los aspectos principales del modelo presentado es que considera simultáneamente unidades funcionales, registros y multiplexores. Se propone una novedosa cota inferior del área de multiplexores que tiene en cuenta la heterogeneidad de los anchos de palabras de sus entradas, así como los diferentes alineamientos posibles entre éstas.

Los resultados muestran que un alto porcentaje del área total se dedica a multiplexores y registros, lo cual revela una falta en las FPGA modernas de estructuras que realicen la multiplexación de datos de forma eficiente. El uso de un conjunto de recursos completo durante el proceso de optimización permite disponer de un modelo arquitectural más exacto que el utilizado por técnicas más tradicionales en las que sólo se incluyen FUs. Se han obtenido mejoras en el área de hasta un 35% (media del 2%) al comparar el modelo completo con el modelo tradicional. El modelo arquitectural es completado definitivamente al incluir también multiplicadores empotrados. Se presenta una función de coste (área) novedosa que permite la selección automática de recursos mediante una distribución eficiente entre recursos basados en LUTs y recursos empotrados. Las altas reducciones obtenidas en el

uso total de recursos no habrían sido posibles si se hubiesen utilizado las funciones de minimización existentes. Los resultados de la comparación entre las implementaciones del tipo UWL y con las del tipo MWL muestran mejoras en área de hasta un 77% (media del 46%) y mejoras en la latencia de un 22% en media tras aplicar compartición de recursos (HLS) en implementaciones homogéneas. Los resultados relativos a implementaciones heterogéneas muestran reducciones de área de hasta un 80 % (media del 44 %) y mejoras en la latencia de un 19% en media. La comparación entre las implementaciones basadas en MWL usando arquitecturas homogéneas y heterogéneas presenta resultados con mejoras en área de hasta un 54% (media del 40%) cuando los recursos empotrados se incluyen en el proceso de optimización.

Resumiendo, las contribuciones principales en lo tocante a las síntesis de alto nivel de sistemas con anchos de palabra múltiples son:

- Se propone una herramienta para el desarrollo de circuitos DSP con MWL optimizados para FPGAs con arquitecturas homogéneas y heterogéneas.
- La herramienta considera FUs, registros y multiplexores simultáneamente.
- Se utiliza una medida de área novedosa (+-norm) para gestionar recursos basados en LUTs y empotrados.
- El enfoque MWL es claramente superior al enfoque UWL.
 - Arquitecturas homogéneas: mejora media del 46%.
 - Arquitecturas heterogéneas: mejora media del 44%.
 - Mejora en la latencia mínima: 22% (caso homogéneo) and 19% (caso heterogéneo)
- El uso de técnicas de selección de módulos para sistemas heterogéneos permiten alcanzar mejoras en el de un 40% en media.

- Los resultados relativos a la implementación de la ruta de datos (*datapath*) muestran que un alto porcentaje de los recursos está dedicado a multiplexores y registros, revelando que existe una deficiencia en las estructuras dedicadas a la multiplexación de datos en dispositivos FPGAs.

El proceso de WLA de sistemas DSP ha sido también abordado centrándose principalmente en dos aspectos: la generación de un estimador del error de cuantificación rápido y preciso, y el análisis del efecto de usar diferentes técnicas de optimización durante la WLA en los resultados de la HLS. La estimación del ruido de cuantificación es esencial para llevar a cabo optimizaciones circuitales complejas, puesto que las técnicas tradicionales basadas en simulación conllevan tiempos de computación extremadamente largos, limitando la complejidad que puede ser aplicada a los algoritmos de optimización. No obstante, los sistemas LTI en régimen permanente son los únicos que permiten un análisis completo que permita obtener estimaciones precisas del ruido de cuantificación. En este trabajo, se propone un estimador de ruido para sistemas LTI y para un conjunto reducido de sistemas no lineales (que incluye algoritmos con realimentaciones). La estimación se basa en el uso de aritmética afín (AA: Affine Arithmetic), puesto que las simulaciones basadas en AA proporcionan información acerca del efecto que cualquier perturbación (por ejemplo, el efecto de la cuantificación) aplicada a una señal produce en el resto de señales del algoritmo, manteniendo información acerca de su señal de origen. Esto hace posible la parametrización de las características estadísticas del error de cuantificación a la salida del algoritmo mediante una función que las relaciona con las propiedades estadísticas de las fuentes de ruido (cuantificación de cada señal). Debido a que el objetivo de esta tesis es combinar dos tareas que por sí solas son complejas, creando una nueva tarea combinada que es aún más compleja, cualquier mejora en los tiempos de optimización es de vital importancia. De aquí, el empeño puesto en el desarrollo de un estimador del error de cuantificación rápido. Este novedoso estimador presenta una precisión elevada en algoritmos LTI y no lineales sin realimentaciones (error

medio de la estimación $<1.2\%$ para el rango de relaciones de señal a ruido de cuantificación (SQNR: Signal to Quantization Noise Ratio) $SQNR = [3, 120]$ dB) y en algoritmos no lineales realimentados (error media $< 3.46 \%$ para $SQNR = [40, 120]$ dB). La aceleración del proceso en comparación con una simulación en coma fija alcanza valores de hasta 1210 veces. Y en lo que se refiere al uso de diferentes técnicas de optimización durante WLA, los resultados muestran que el comportamiento obtenido cuando la arquitectura es totalmente paralela (sin compartición de recursos) no se mantiene tras la aplicación posterior de HLS, en la que sí se comparten recursos. Por ejemplo, si la arquitectura es paralela las técnicas basadas en SA ofrecen mejores resultados que técnicas basadas en el descenso de gradiente. Sin embargo, si se aplican las técnicas de WLA y HLS secuencialmente, en algunos casos no existe diferencia entre los resultados de las distintas técnicas de optimización de anchos de palabras, e incluso es posible que las técnicas basadas en el descenso de gradiente superen a las basadas en recocido simulado. Los resultados indican que la aplicación secuencial de WLA y HLS introduce incertidumbre acerca de la calidad de los métodos de optimización de anchos de palabra, probando la necesidad de la aplicación combinadas de estas tareas de diseño. Sin embargo, en media, las técnicas de optimización de anchos de palabra complejas, obtienen mejores resultados que técnicas más básicas, como el descenso de gradiente. Como ejemplo, SA obtiene una mejora media en el área del 5% para implementaciones homogéneas y del 4,75% en implementaciones heterogéneas.

En resumen, la principales contribuciones relativas a técnicas de WLA son:

- Se presenta una técnica novedosa de estimación del ruido de cuantificación:
 - Aplicable a algoritmos LTI y a un subconjunto de algoritmos no-lineales que incluyen transitorios.

- Alta precisión para sistemas LTI y no lineales sin realimentación: error medio de la estimación $< 2\%$.
 - Alta precisión para sistemas no lineales realimentados: error medio de la estimación $< 4\%$.
 - Reducción de los tiempos de estimación de hasta 1210 veces.
- Los métodos actuales de optimización aplicados en la WLA no tienen en cuenta determinados aspectos arquitecturales y, por lo tanto, producen resultados subóptimos.
 - Las prestaciones de las técnicas de WLA no se mantienen tras la aplicación de HLS: técnicas subóptimas pueden obtener mejores resultados que técnicas cuasi-óptimas.
 - Técnicas de WLA basadas en SA obtienen una mejora media en el área del 5% comparada con técnicas basadas en el descenso de gradiente.

La aplicación conjunta de la WLA y la HLS ha sido abordada utilizando dos enfoques distintos: un enfoque óptimo y otro heurístico. El enfoque óptimo se desarrolla mediante el uso de programación matemática (MILP: Mixed Integer Linear Programming). Debido a los tiempos de convergencia extremadamente largos que se necesitan para resolver el problema óptimo, ha sido necesario simplificar el problema original además de hacer uso de ejemplos de complejidad pequeña. Los resultados de la resolución óptima del problema combinado han sido comparados con la aplicación secuencial de técnicas de WLA y de HLS basadas a su vez en MILP, es decir, técnicas óptimas. Los resultados obtenidos arrojan algo de luz sobre los posibles pasos a seguir en el desarrollo de un heurístico que combine la WLA y la HLS. El problema combinado implica la variación simultánea de los tamaños de los anchos de palabras, del conjunto de recursos y también de la asignación entre operaciones y recursos.

El análisis óptimo muestra que se obtienen beneficios al aplicar el enfoque combinado, pero de forma esporádica. Tan sólo determinadas combinaciones de restricciones de latencia y error a la salida producen mejoras. Las mejoras obtenidas son de hasta un 13%, lo cual valida el enfoque combinado.

Los resultados de la resolución óptima del problema combinado mediante MILP han permitido seleccionar un conjunto de *movimientos* que pueden ser aplicados durante SA y que han sido utilizados para desarrollar el algoritmo heurístico que aborda el problema combinado. Estos movimientos se basan en movimientos tradicionales que afectan a la asignación de recursos, movimientos tradicionales de cuantificación, y finalmente movimientos específicos que combinan ambas tareas. Como resultado, se ha propuesto un algoritmo de síntesis que es capaz de optimizar implementaciones DSP con restricciones de latencia y error a la salida. De nuevo, el conjunto de recursos está compuesto por FUs, registros y multiplexores, y también incluye recursos empotrados.

La propuesta heurística ha sido comparada con la ejecución secuencial de técnicas WLA y HLS basadas también en SA. La comparación indica que la propuesta combinada produce mejoras en el área para la mayoría de los experimentos realizados. Las mejoras obtenidas en implementaciones homogéneas son en media del 4% alcanzando un máximo del 21%. La implementación usando arquitecturas heterogéneas añade una nueva dimensión al problema, ya que las mejoras pueden ser medidas de dos formas: en relación al número de veces que la implementación se puede replicar en la FPGA (inversamente proporcional a la norma infinito del vector de área), o en relación al uso total de recursos cuando las normas infinito coinciden. El número de veces que un circuito puede ser replicado mejora en media un 4.5% alcanzando un máximo del 18%. El uso total de recursos ha sido mejorado un 5.6% en media, alcanzado un máximo del 10%. El análisis de los resultados sugiere la existencia de una tendencia de las mejoras a incrementarse conforme la complejidad de los algoritmos crece. Tal como ya se ha mencionado, una de las fases siguientes a este trabajo de investigación está centrada

en la optimización de la eficiencia computacional de las herramientas de síntesis de manera que algoritmos complejos puedan ser implementados en tiempos acotados, para así poder confirmar las ventajas del uso del enfoque combinado para sistemas industriales.

Las contribuciones principales y conclusiones con respecto al caso combinado son:

- Se ofrece una descripción basada en MILP del problema combinado.
- Los resultados óptimos muestran que las mejoras aparecen de forma esporádica, obteniéndose un máximo del 13%. Estos resultados se han usado para extraer posibles estrategias aplicables a algoritmos heurísticos de optimización.
- Una herramienta completa que desarrolla circuitos DSP optimizados basados en MWL para arquitecturas homogéneas y heterogéneas capaz de seleccionar simultáneamente el ancho de palabras de las señales y la arquitectura de la implementación.
- La herramienta considera FUs, registros y multiplexores simultáneamente.
- La herramienta considera recursos basados en LUTs y recursos empotrados.
- El enfoque combinado proporciona mejoras para la mayoría de los casos.
- Las implementaciones homogéneas se mejoran en un 4% (máximo del 21%).
- La arquitecturas heterogéneas presentan:
 - una mejora media del número de veces que puede replicarse la implementación del 4.5% (máximo del 18%).
 - una mejora media del uso total de recursos del 5.6% (máximo del 10%).
- Los resultados sugieren que las mejoras obtenidas aumenten conforme la complejidad de los algoritmos DSP crece.

PUBLICACIONES

Las contribuciones de esta Tesis, así como otros trabajos de investigación relacionados que han sido llevados a cabo, han sido publicados en las actas de congresos internacionales y en revistas internacionales. A continuación presentamos brevemente dichas publicaciones explicando cuáles han sido las aportaciones al estudio de la combinación de la WLA y la HLS.

Asignación de Anchos de Palabra de sistemas DSP

Las publicaciones sobre WLA incluyen trabajos que cubren gran parte del espectro de posibilidades que forman esta línea de investigación. Se han abordado tanto técnicas analíticas como técnicas basadas en simulación, algoritmos lineales y no lineales, diferentes técnicas de redondeo, ciclos límite, etc. Para llevar acabo estos trabajos ha sido necesario el desarrollo en el lenguaje C/C++ de herramientas propias que han sentado las bases del entorno software utilizado durante esta Tesis.

- G. Caffarena, J.A. López, A. Fernandez, C. Carreras, and O. Nieto-Taladriz, "Quantization Noise Estimator for DSP Algorithms", en preparación para *IEEE Transactions on Circuits and Systems-I*.
- J.A. López, G. Caffarena, C. Carreras, and O. Nieto-Taladriz, "Fast and Accurate Computation of the Round-Off Noise of LTI Systems", aceptado en *IET Circuit, Devices and Systems*, 2008.
- G. Caffarena, A. Fernández, C. Carreras, and O. Nieto-Taladriz, "Fixed-point refinement of OFDM-based adaptive equalizers: A heuristic approach", en *European Signal Processing Conference*, pp. 1353-1356, 2004.

- J.A. López, G. Caffarena, C. Carreras, and O. Nieto-Taladriz, "Analysis of limit cycles by means of affine arithmetic computer-aided tests", en *European Signal Processing Conference*, pp. 991-994, 2004.
- J.A. López, G. Caffarena, C. Carreras, and O. Nieto-Taladriz, "Characterization of the quantization properties of similarity-related DSP structures by means of interval simulations", en *IEEE Asilomar Conf. on Signals, Systems and Computers*, vol. 2, pp. 2208-2212, 2003.
- J.A. López, C. Carreras, G. Caffarena, and O. Nieto-Taladriz, "Fast characterization of the noise bounds derived from coefficient and signal quantization", en *IEEE Int. Symp. on Circuits and Systems*, vol. 4, pp. 309-312, 2003.

Síntesis de Alto Nivel de sistemas DSP

Se han desarrollado técnicas de HLS para sistemas DSP que explotan las posibilidades del uso optimizado de *cores* aritméticos basados en LUTs y *cores* empotrados. Dichas técnicas han sido una de las contribuciones más importantes de este trabajo de investigación puesto que han sido la semilla a partir de la cual se han generado las técnicas de combinación de WLA y HLS. El nuevo enfoque propuesto permite la implementación de sistemas DSP muy optimizados, ya que es posible sacar el máximo provecho a las posibilidades que ofrecen los dispositivos FPGA actuales.

- G. Caffarena, J.A. López, C. Carreras, and O. Nieto-Taladriz, "High-Level Synthesis of Multiple Word-Length DSP Algorithms using Heterogeneous-Resource FPGAs", en *Field Programmable Logic and Applications*, pp.675-678, 2006.
- G. Caffarena, J. A. López, C. Carreras, and O. Nieto-Taladriz, "Optimized implementation of DSP cores on FPGAs using logic-based and embedded resources," en *Symp. on System-on-Chip*, pp.103-106, 2006.

Aplicación conjunta de la Asignación de Anchos de Palabra y la Síntesis de Alto Nivel de sistemas DSP

La combinación de WLA y HLS se desarrolla en los artículos expuestos más abajo.

La primera publicación se basa en un enfoque óptimo al problema combinado, y tiene como objetivo principal probar la validez de la propuesta de combinación y sentar las bases para la implementación de algoritmos heurísticos que sean capaces de abordar el problema que nos ocupa. Los resultados obtenidos han sido utilizados como punto de partida para el desarrollo de un entorno de síntesis de sistemas DSP que saca provecho de los beneficios que otorgan la combinación de WLA con HLS. En este artículo, el problema combinado se ha formulado en su totalidad por primera vez, puesto que anteriores publicaciones sólo han abordado el problema parcialmente.

También se ha desarrollado un enfoque heurístico. La propuesta presenta una serie de algoritmos que permite realizar la síntesis de circuitos en su totalidad, cubriendo todas las fases pertinentes. Dicha propuesta se basa en el intercambio entre ruido de cuantificación y el uso de recursos para así optimizar el coste de implementación final. La propuesta considera tanto dispositivos FPGA homogéneos como heterogéneos. Las mejoras obtenidas por esta novedosa aproximación al problema de síntesis alcanzan un 20% para ambos tipos de arquitecturas.

- G. Caffarena, G. Constantinides, P. Cheung, C. Carreras, and O. Nieto-Taladriz, "Optimal combined word-length allocation and architectural synthesis of digital signal processing circuits", *IEEE Trans. Circuits Syst. II*, vol. 53, pp. 339-343, May 2006.
- G. Caffarena, C. Carreras, and O. Nieto-Taladriz, "Noise-Aware High-Level Synthesis of Digital Signal Processing Algorithms using Heterogenous-Architecture FPGAs", in preparación para *IEEE Trans. on Computer-Aided Design*.

Implementación de sistemas de procesamiento de alta velocidad

Se han desarrollado varios sistemas de computación de altas prestaciones basados en dispositivos FPGA. Estos trabajos, que tienen naturaleza más práctica, han aportado información muy valiosa acerca de la aplicación práctica de WLA a sistemas complejos, así como acerca del impacto en las prestaciones finales de los diseños que tiene el disponer de un alto conocimiento de las arquitecturas de los dispositivos FPGA y de las herramientas CAD comerciales. Las publicaciones son las siguientes:

Las publicaciones son:

- G. Caffarena, C. Pedreira, C. Carreras, S. Bojanic, O. Nieto-Taladriz, "FPGA Acceleration for DNA Sequence Alignment", *Journal of Circuits, Systems and Computers*, vol. 16(3), pp. 245-266, April 2007
- S. Bojanic, G. Caffarena, S. Petrovic, O. Nieto-Taladriz. , "FPGA for pseudorandom generator cryptanalysis", *Journal of Microprocessors and Microsystems*, vol. 30(2), pp.63-71, May 2006.
- G. Caffarena, A. Fernández, C. Carreras, J.A. López, O. Nieto-Taladriz, "Design of OFDM-based Adaptive Equalizer for WLAN: Implementation Issues", *Mediterranean Journal of Electronics and Communication*, vol 1(1), pp. 25-34, Oct. 2005.
- S. Bojanic, G. Caffarena, S. Petrovic, O. Nieto-Taladriz, "Stream cipher cryptanalysis based on the edit-distance: A hardware approach", *Tatra Mountains Mathematical Publications*, vol. 29 (2), pp. 17-29, 2004.
- A. Fernández, A. Jiménez, G. Caffarena, J. Casajús, "Design and implementation of a hardware module for equalisation in a 4G MIMO receiver", *Int. Conf. on Field Programmable Logic and Applications*, pp. 675-678, 2006.
- G. Caffarena, S. Bojanic, J.A. López, C. Pedreira, O. Nieto-Taladriz, "High-speed

systolic array for gene matching", *ACM/SIGDA 12th Int. Symp. on Field Programmable Gate Arrays*, pp. 248, 2004

- G. Caffarena, S. Bojanic, J.A. López, C. Pedreira and O. Nieto-Taladriz, "Parallel computation of Gene Sequence Matching", *IADIS Int. Conf. on Applied Computing*, pp. 18-25, 2004.
- C. Pedreira, G. Caffarena, S. Bojanic, V. Pejovic, O. Nieto-Taladriz, "Incrementado la velocidad de algoritmos de búsqueda basados en FPGAs", *XIV Jornadas Telecom I+D*, Madrid (Spain), 2004.
- S. Bojanic, G. Caffarena, C. Pedreira, O. Nieto-Taladriz, "High Speed Circuits for Genetics Applications", *Int. Conf. on Microelectronics*. pp. 517-524, 2004, Invited Keynote Paper.
- S. Bojanic, G. Caffarena, S. Petrovic, and O. Nieto-Taladriz, "Stream Cipher Sequence Matching using Reconfigurable Hardware", *IASTED International Conference on Communication Systems and Networks*, pp.62-67, September 2003.

Diseño de bloques aritméticos

La generación automática de bloques aritméticos de coma fija y coma flotante para dispositivos FPGA, así como el desarrollo de modelos de coste (área, latencias y consumo de potencia) constituyen otra de las contribuciones clave de esta Tesis, puesto que han permitido al autor evaluar el impacto que tiene utilizar una librería de componentes de alta calidad en el diseño de sistemas DSP, mediante técnicas manuales y automáticas (HLS). De nuevo, el desarrollo de software basado en C/C++ para realizar la exploración del espacio de diseño y la generación automática de código VHDL ha sido de gran utilidad.

- R. Jevtic, C. Carreras and G. Caffarena, "Fast and Accurate Power Estimation of FPGA

DSP Components Based on High-level Switching Activity Models", aceptado en *Int. Journal of Electronics*, 2008.

- J.A. López, G. Caffarena, C. Carreras, O. Nieto-Taladriz, "Optimizing the Hardware Implementation of Constant Coefficient Dividers Using the Properties of Quantization Operations ", *WSEAS Transactions on Circuits and Systems*, vol. 4(5), 462-480, May 2005.
- R. Jevtic and G. Carreras, C. Caffarena, "Switching Activity Models for Power Estimation in FPGA Multipliers", *Int. Workshop on Applied Reconfigurable Computing*, pp. 201-213, 2007.
- R. Jevtic, C. Carreras and G. Caffarena, "High-level Switching Activity Models for Multipliers in FPGAs", *ACM/SIGDA Int. Symp. on Field-Programmable Gate Arrays*, pp. 224-225, 2007.
- G. Leyva, G. Caffarena, C. Carreras, O. Nieto-Taladriz, "A generator of high-speed floating-point modules", *IEEE Symp. on Field-Programmable Custom Computing Machines*, pp. 306-207, 2004.
- G. Leyva, G. Caffarena, C. Carreras, O. Nieto-Taladriz, "Generation of High-Speed Parameterized Floating-Point Units", *Int. Conf. on Reconfigurable Computing and FPGAs*, pp. 294-300, 2004, Quality paper award.

CHAPTER 1

INTRODUCTION

Digital Signal Processing (DSP) has a major role in communication and multimedia systems. The ever increasing demand for application performance can only be fulfilled by the concurrent development of both microelectronic technology and DSP techniques. For instance, modern wireless communication systems require the processing of several hundreds of high-speed data channels, and this fact highly complicates the system design, since both the DSP algorithms and their electronic implementations are being pushed to the limit.

The high complexity of these digital systems requires the introduction of powerful Computer-Aided Design (CAD) tools to make their physical implementation possible. Any improvement in the algorithms implemented by CAD tools is always welcome, since it may lead to design cost reductions. In fact, the Electronic Design Automation (EDA) companies are continuously evolving, trying to keep the pace of users' demands for more complex applications and to make the most of microelectronic advances. It is in this line, the development of a new generation of design tools adapted in particular to Field Programmable Gate Array (FPGA) based technologies, that this thesis is focused on.

The design flow is divided into several tasks that allow the designer to traverse the path from the design specification to the implementation. The analysis of the possible combination of two well-known design tasks, Word-Length Allocation (WLA) and High-Level Synthesis (HLS), applied to the generation of specific DSP hardware architectures, is the main topic of this research work, having as a goal the improvement of current DSP design techniques for FPGAs.

The chapter is organized as follows: The motivations of this thesis are presented in section 1.1. Next, the objectives of the thesis are enumerated. Section 1.3 presents the organization of the thesis. Section 1.4 exposes the contributions of this thesis. Finally, the publications in international congresses and journals that this thesis has produced are listed in section 1.5.

1.1. Motivation

Word-length allocation, also called quantization, is the translation of an initial infinite-precision description of an algorithm into a finite-precision description. Fixed-point arithmetic is commonly used to implement DSP circuits, as it has proved to enable low-cost, low-power and high-speed implementations, compared to implementations using floating-point arithmetic. Therefore, in this work the term quantization implies the translation of an infinite-precision description of an algorithm (it is normally assumed that the floating-point format IEEE 754-1984 suffices for that) into a fixed-point description, meeting a given error constraint.

It is necessary to wisely select the word-lengths of the different signals composing the hardware implementation in order to minimize design costs. Traditionally, a Uniform Word-Length (UWL) approach, inherited from the microprocessor-based design point of view, has been used to produce fast quantization procedures, but clearly wasting hardware resources.

In the last two decades, this approach has turned into the more powerful Multiple Word-Length (MWL) approach, which customizes the word-length of each signal, thus reducing design costs dramatically. However, the MWL paradigm implies more complex quantization and HLS algorithms.

HLS' final goal is to explore the design space to find the best hardware architecture that complies with the power, area and speed requirements and constraints. It is mainly composed of the scheduling, resource allocation and resource binding phases. HLS transforms an initial algorithm specification (i.e. a Data Flow Graph (DFG)) into a Register Transfer Level (RTL) description, while meeting certain design constraints.

The tasks of WLA and HLS are traditionally carried out separately to reduce computational complexity. First, WLA is performed to obtain the word-lengths of the algorithm variables. Then, HLS analyzes the description of the quantized algorithm and produces the final architecture to be implemented. The main drawback of this methodology is that it does not take into account the interdependencies between these two tasks. Word-lengths are normally reduced during WLA until the upper noise bound is met. In this stage it is common to ignore architectural issues, and this reduces the problem to simpler terms. For instance, a design goal can be to minimize the total number of variables' bits, as that might reduce the area of arithmetic operators, the amount of wiring, etc. There are some approaches where an architectural model is introduced, and this model is used to guide the selection of word-lengths. However, this model assumes a fully parallel implementation of the algorithm, without considering the architectural model used during HLS. Hence, the final cost of the system after HLS is not guaranteed to be minimal, since the architectural models used in both tasks may differ considerably.

The combined application of both design tasks would allow the use of a single architectural model during the whole design process, thus leading to highly optimized implementations, since a more thorough design space exploration is carried out. The cost to pay is the

generation of new techniques to address the combined problem, and due to the fact that both WLA and HLS are already quite complex tasks, the combination of both has a much higher complexity. The latter fact might be easily overcome in part by means of the continuous advances in computing power, which allow engineers and scientists to face more and more complex problems everyday. However, new techniques to efficiently address the combined problem are still required. As of today, the combined approach has been tackled just by a few research groups, and only superficially, so there are still many open aspects to be researched.

Another important fact is the correct adaptation to the target technology under consideration. Programmable devices, and particularly Field-Programmable Gate Arrays, have gained special relevance in the past few years. Their features place them at the border between Application Specific Integrated Circuits (ASICs) and microprocessor-based systems. Basically, they allow the implementation of dedicated architectures that can be reconfigured. Their computing performance is in between that of ASICs and microprocessors. Their reconfiguration capabilities make FPGAs a suitable prototyping platform. However, their recently achieved high performance features (high level of integration and high-speed processing) make them also an interesting choice for final product implementations. The academic world has welcomed them due to both the possibility they offer to reuse the same device to test different hardware systems, and the power of the available design tools. For these reasons, FPGAs have been chosen as the target technology for this work. The recent advances in FPGAs, such as the introduction of specialized embedded blocks, offer new possibilities for the implementation of high-performance DSP algorithms. Numerous efforts are being made to find architectural models, as well as automatic design algorithms, which allow the exploitation of the capabilities that FPGAs offer. For instance, there are approaches which propose moving from the homogeneous-resource architecture paradigm (i.e. only look-up table (LUT) based FPGA resources) to the more complex, yet more powerful, heterogeneous-resource architecture paradigm (i.e. LUT-based as well as specialized embedded FPGA resources). This

ambitious approach will also be considered in the work presented here.

1.2. Objectives

The main objectives of this thesis are:

- In depth review of the state of the art of WLA, HLS and the combined application of both tasks.
- Generation of a complete HLS method able to handle:
 - MWL DSP algorithms
 - FUs, storing resources and steering logic simultaneously
 - heterogeneous FPGA resources (LUT-based and embedded resources)
- Comparative analysis of the HLS results obtained using homogeneous and heterogeneous FPGA architectures.
- Generation of a fast and accurate quantization error model for both LTI and non-linear systems.
- Analysis of the effect of the architectural model used during WLA on HLS.
- Analysis of the effect of different optimization methods during WLA on HLS.
- Optimal analysis of the problem that combines WLA and HLS based on linear programming techniques.
- Generation of a heuristic synthesis algorithm able to combine WLA and HLS.
- Analysis of the results of the combined WLA and HLS for homogeneous FPGA architectures with respect to the traditional approach (sequential application of WLA and HLS).

- Analysis of the results of the combined WLA and HLS for heterogeneous FPGA architectures.

1.3. Book organization

The remaining chapters of this thesis are organized as follows: Chapter 2 deals with the state of the art in related research topics such as WLA of MWL, HLS of MWL, the combined application of WLA and HLS, and FPGA technological issues. Chapter 3 addresses HLS. A HLS framework suitable for implementations of MWL DSP systems in heterogeneous-architecture FPGAs is presented as well as the related implementation results. In the next chapter, WLA is tackled. A novel error estimation method as well as the analysis of the impact of WLA on the implementation are presented. Chapter 5 proposes methods and provide with results on the combined application of WLA and HLS. Both optimal and heuristic methods are presented. Finally, chapter 6 contains the conclusions of this work.

1.4. Contributions of the thesis

While pursuing the combination of WLA and HLS it has been necessary to analyze each task separately, and also to attack the problem of the concurrent application of both, bearing in mind that the target technology was FPGA technology. As a result, this thesis has contributed in the fields of WLA, HLS using FPGAs, and eventually in the combined application of WLA and HLS for implementation in FPGAs.

A complete approach of HLS has been implemented. The proposal enables the consideration of architectural issues which are normally neglected (i.e. the simultaneous use of FUs, registers and multiplexers), as well as heterogeneous FPGA resources (i.e. LUT-based and embedded resources), and it is also suitable for MWL DSP implementations. The approach makes use of a resource library that accounts for MWL effects for FUs, registers

and multiplexers, thus producing highly optimized architectures. This library includes both LUT-based and embedded FPGA resources, which further increase the power of the HLS task.

Another important contribution is the introduction of resource usage metrics suitable for heterogeneous-architecture FPGAs within the HLS procedure. As a result, a novel HLS procedure able to distribute resource usage between LUT-based and embedded resources, while optimizing the overall FPGA resource usage is presented. A novel resource usage metric is proposed, which outperforms previous approaches in terms of computational efficiency and optimization results.

A novel quantization error estimation based on Affine Arithmetic (AA) is presented, as well as its practical application to the automatic WLA of LTI and non-linear differentiable DSP systems. The error estimation is based on performing a pre-processing of the DFG representing the algorithm, which produces an expression of the quantization error at the system output. This error only depends on the word-lengths of the algorithm variables, thus allowing accurate and fast automatic WLA. The error estimation method is more accurate and more computationally efficient than previously existing approaches.

The analysis of the impact of different optimization techniques during WLA on HLS results is also presented. The variance in the obtained results corroborates the fact that it is worth using a single architecture model during WLA and HLS, and this is only possible by means of combining these tasks.

The actual combination of WLA and HLS has been first performed by using a Mixed Integer Linear Programming (MILP) approach. The results prove the validity of the approach and also provide with insights into the combination of the two tasks, which are of most significance to generate heuristic synthesis algorithms able to cope with industrial DSP systems.

Finally, the global contribution of this thesis is an HLS heuristic algorithm able to per-

form the combined WLA and HLS of DSP systems for both homogeneous and heterogeneous FPGA architectures. Up to 20% of resource usage reduction is reported, which proves the importance of such a combined approach, providing electronic designers with a design framework that enables highly improved DSP custom hardware implementations.

1.5. Publications

This thesis' contributions along with other related research performed, have been published in international conferences and journals, and it is the goal of this section to briefly present them and to expose their contribution to the study of the combination of WLA and HLS.

Word-length allocation of DSP systems

The publications on WLA comprise works that range from analytical techniques to simulation-based techniques. They also deal with both LTI and non-linear systems, as well as with rounding noise and limit cycles estimation. These works include the development of in-house C/C++ tools that have settled the grounds for the software framework developed during this thesis.

- G. Caffarena, J.A. López, A. Fernandez, C. Carreras, and O. Nieto-Taladriz, "Quantization Noise Estimator for DSP Algorithms", in preparation for *IEEE Trans. on Circuits and Devices I*.
- J.A. López, G. Caffarena, C. Carreras, and O. Nieto-Taladriz, "Fast and Accurate Computation of the Round-Off Noise of LTI Systems", to be published in *IET Circuit, Devices and Systems*, 2008.
- G. Caffarena, A. Fernández, C. Carreras, and O. Nieto-Taladriz, "Fixed-point refinement of OFDM-based adaptive equalizers: A heuristic approach", in *European Signal Processing Conference*, pp. 1353-1356, 2004.

- J.A. López, G. Caffarena, C. Carreras, and O. Nieto-Taladriz, "Analysis of limit cycles by means of affine arithmetic computer-aided tests", in *European Signal Processing Conference*, pp. 991-994, 2004.
- J.A. López, G. Caffarena, C. Carreras, and O. Nieto-Taladriz, "Characterization of the quantization properties of similarity-related DSP structures by means of interval simulations", in *IEEE Asilomar Conf. on Signals, Systems and Computers*, vol. 2, pp. 2208-2212, 2003.
- J.A. López, C. Carreras, G. Caffarena, and O. Nieto-Taladriz, "Fast characterization of the noise bounds derived from coefficient and signal quantization", in *IEEE Int. Symp. on Circuits and Systems*, vol. 4, pp. 309-312, 2003.

High-level synthesis of DSP systems

The HLS of DSP systems exploiting the capabilities of both LUT-based arithmetic cores and embedded multipliers, has been one of the most important contributions of this thesis, since it has been the seed from which the final combined WLA and HLS approach has been developed. This new approach to HLS allows the implementation of highly optimized DSP systems, since it is possible to fully exploit the capabilities and resources of modern FPGAs.

- G. Caffarena, J.A. López, C. Carreras, and O. Nieto-Taladriz, "High-Level Synthesis of Multiple Word-Length DSP Algorithms using Heterogeneous-Resource FPGAs", in *Field Programmable Logic and Applications*, pp.675-678, 2006.
- G. Caffarena, J. A. López, C. Carreras, and O. Nieto-Taladriz, "Optimized implementation of DSP cores on FPGAs using logic-based and embedded resources," in *Symp. on System-on-Chip*, pp.103-106, 2006.

Combined word-length allocation and high-level synthesis of DSP systems

The actual combination of WLA and HLS is presented in the papers below.

In the first publication, an optimal approach is taken, and it is the main goal of this work to prove the validity of the combined approach and to set the grounds for the implementation of heuristic algorithms able to cope with the problem. The results provided have been used as a starting point to develop a DSP synthesis framework which exploits the benefits of combining WLA and HLS. In this work, the actual combination of WLA and HLS is formulated for the first time, since previous approaches only tackled the problem partially.

A heuristic approach is also developed. A complete HLS approach able to trade-off quantization noise with resource usage is presented. The proposed method automatically synthesizes an algorithm complying with given latency and output precision constraints. The approach is suitable for both homogeneous and heterogeneous FPGA architectures. The improvement in resource usage obtained by this novel approach is up to 20% for homogenous and heterogeneous architectures.

- G. Caffarena, G. Constantinides, P. Cheung, C. Carreras, and O. Nieto-Taladriz, "Optimal combined word-length allocation and architectural synthesis of digital signal processing circuits", *IEEE Trans. Circuits Syst. II*, vol. 53, pp. 339-343, May 2006.
- G. Caffarena, C. Carreras, and O. Nieto-Taladriz, "Noise-Aware High-Level Synthesis of Digital Signal Processing Algorithms using Heterogeneous-Architecture FPGAs", in preparation for *IEEE Trans. on CAD*.

Implementation of high-speed processing systems

Several high-performance FPGA-based computation systems have also been implemented. These more practical works have provided valuable information about the practical application of WLA to complex systems and also about the impact on the design

performance of the knowledge of FPGA architecture issues and commercial computer-aided design tools. The publications are:

- G. Caffarena, C. Pedreira, C. Carreras, S. Bojanic, O. Nieto-Taladriz, "FPGA Acceleration for DNA Sequence Alignment", *Journal of Circuits, Systems and Computers*, vol. 16(3), pp. 245-266, April 2007
- S. Bojanic, G. Caffarena, S. Petrovic, O. Nieto-Taladriz. , "FPGA for pseudorandom generator cryptanalysis", *Journal of Microprocessors and Microsystems*, vol. 30(2), pp.63-71, May 2006.
- G. Caffarena, A. Fernández, C. Carreras, J.A. López, O. Nieto-Taladriz, "Design of OFDM-based Adaptive Equalizer for WLAN: Implementation Issues", *Mediterranean Journal of Electronics and Communication*, vol 1(1), pp. 25-34, Oct. 2005.
- S. Bojanic, G. Caffarena, S. Petrovic, O. Nieto-Taladriz, "Stream cipher cryptanalysis based on the edit-distance: A hardware approach", *Tatra Mountains Mathematical Publications*, vol. 29 (2), pp. 17-29, 2004.
- A. Fernández, A. Jiménez, G. Caffarena, J. Casajús, "Design and implementation of a hardware module for equalisation in a 4G MIMO receiver", *Int. Conf. on Field Programmable Logic and Applications*, pp. 675-678, 2006.
- G. Caffarena, S. Bojanic, J.A. López, C. Pedreira, O. Nieto-Taladriz, "High-speed systolic array for gene matching", *ACM/SIGDA 12th Int. Symp. on Field Programmable Gate Arrays*, pp. 248, 2004
- G. Caffarena, S. Bojanic, J.A. López, C. Pedreira and O. Nieto-Taladriz, "Parallel computation of Gene Sequence Matching", *IADIS Int. Conf. on Applied Computing*, pp. 18-25, 2004.

- C. Pedreira, G. Caffarena, S. Bojanic, V. Pejovic, O. Nieto-Taladriz, "Incrementado la velocidad de algoritmos de búsqueda basados en FPGAs", *XIV Jornadas Telecom I+D*, Madrid (Spain), 2004.
- S. Bojanic, G. Caffarena, C. Pedreira, O. Nieto-Taladriz, "High Speed Circuits for Genetics Applications", *Int. Conf. on Microelectronics*. pp. 517-524, 2004, Invited Keynote Paper.
- S. Bojanic, G. Caffarena, S. Petrovic, and O. Nieto-Taladriz, "Stream Cipher Sequence Matching using Reconfigurable Hardware", *IASTED International Conference on Communication Systems and Networks*, pp.62-67, September 2003.

Design of arithmetic cores

The automatic generation and the cost modeling (e.g. area, latency and power consumption) of fixed-point and floating-point arithmetic FPGA cores has also been a key issue in the contributions of this thesis, since it has allowed the author to assess the impact of using a high-quality component library during the design of DSP systems, by both manual and automatic (i.e. HLS) design techniques. Again, the development of C/C++ software to perform design space exploration and automatic VHDL generation has been highly fruitful.

- R. Jevtic, C. Carreras and G. Caffarena, "Fast and Accurate Power Estimation of FPGA DSP Components Based on High-level Switching Activity Models", to be published in *Int. Journal of Electronics*, 2008.
- J.A. López, G. Caffarena, C. Carreras, O. Nieto-Taladriz, "Optimizing the Hardware Implementation of Constant Coefficient Dividers Using the Properties of Quantization Operations ", *WSEAS Transactions on Circuits and Systems*, vol. 4(5), 462-480, May 2005.

- R. Jevtic and G. Carreras, C. Caffarena, "Switching Activity Models for Power Estimation in FPGA Multipliers", *Int. Workshop on Applied Reconfigurable Computing*, pp. 201-213, 2007.
- R. Jevtic, C. Carreras and G. Caffarena, "High-level Switching Activity Models for Multipliers in FPGAs", *ACM/SIGDA Int. Symp. on Field-Programmable Gate Arrays*, pp. 224-225, 2007.
- G. Leyva, G. Caffarena, C. Carreras, O. Nieto-Taladriz, "A generator of high-speed floating-point modules", *IEEE Symp. on Field-Programmable Custom Computing Machines*, pp. 306-207, 2004.
- G. Leyva, G. Caffarena, C. Carreras, O. Nieto-Taladriz, "Generation of High-Speed Parameterized Floating-Point Units", *Int. Conf. on Reconfigurable Computing and FPGAs*, pp. 294-300, 2004, Quality paper award.

CHAPTER 2

STATE OF THE ART

The tasks of WLA and HLS have been extensively researched in recent decades, and it is only in the last decade that their interdependence has been addressed, yet only vaguely. There have been studies regarding the effect of a customized WLA on the final architecture of the system, but constrained to very specific terms (i.e. datapaths with no resource sharing). Also, there are new ways to approach HLS that move apart from the simplistic UWL point of view, which improve the quality of the systems. However, due to the high complexity of the MWL-HLS task, there are still many word-length related open issues (i.e. MWL register binding, MWL steering logic, MWL data storage, etc). Regarding the actual combination of WLA and HLS there are only a few published works, being the results only partial since the combination of these tasks is performed under highly simplified assumptions. Another important fact is the adaptation of WLA and HLS to the target technology (i.e. FPGAs). The recent evolution in FPGA architectures calls for a redesign of WLA and HLS specially focused on the way that resources are modeled. The emergence of specialized DSP and memory blocks in FPGAs offers an interesting opportunity to improve performance

by means of design techniques able to exploit their capabilities. For instance, embedded fast multipliers present in most modern FPGA enable the implementation of high-precision and high-speed systems, but this can only be ultimately achieved by means of new high quality synthesis methods. All these issues are reviewed in this chapter, which is aimed at setting the background for the research work in view.

The chapter is organized as follows: Section 2.1 revises the state of the art on word-length allocation. Section 2.2 addresses the high-level synthesis of DSP algorithms. Next, the works on the combination of WLA and HLS are analyzed in section 2.3. In section 2.4, reconfigurable-logic devices' issues are commented. Finally, the conclusions are drawn in section 2.5.

2.1. Word-length allocation

The original infinite precision of an algorithm based on the use of real arithmetic, must be reduced to the practical precision bounds imposed by digital computing systems. WLA aims at the selection of the word-lengths of variables of an algorithm to comply with a certain output's noise constraint while optimizing the characteristics of the implementation (e.g. area, speed or power consumption). Normally, the precision loss committed is computed by using a double precision floating-point arithmetic (i.e. IEEE 754) description of the algorithm used as a reference and, although there are some works on quantization for custom floating-point arithmetic [FCR03, GML⁺02, GML04], the common approach is to implement the system using fixed-point arithmetic, since this leads to low cost implementations in terms of area, speed and power consumption [CVDM88, Con03, HKBR06, SK95].

Table 2.1 contains a summary of the information presented in the remaining of this subsection. The following acronyms and abbreviations are used:

Analytical (A)	Simulation-based (S)
Interval Arithmetic (IA)	Multi-Interval Arithmetic (MIA)
Affine Arithmetic (AA)	Quantized Affine Arithmetic (QAA)
Transfer Function Analysis (TBA)	Forward and Backward Propagation (FBP)
Optimal (O)	Heuristic (H)
Integer Linear Programming (ILP)	Exhaustive Search (EXH)
Simulated Annealing (SA)	Adaptive Simulated Annealing (ASA)
Linear Time-Invariant (LTI)	Nonlinear (NL)

In this work, only the effect of quantizing signals is considered, despite the fact that WLA has a wider scope, since it also includes the analysis of the effect of coefficient quantization [Liu71, OW72] and the analysis of limit cycles effects [RM87, LCCNT04]. The quantization of signals introduces two possible error sources: overflow errors, due to the quantization of the most significant bits (MSB) of signals, and underflow errors due to the quantization of the least significant bits (LSB). Overflow error can be avoided, or at least highly minimized, by means of computing the scaling (i.e. position of the fractionary point) of each signal of the algorithm, or, alternatively, it can be handled by making use of signal saturation techniques. Underflow noise, however, cannot be avoided as it is always necessary to use a finite number of bits to represent signals and an infinite number of bits is required to represent certain numbers (i.e. rational numbers). There are basically two main approaches to LSB quantization: rounding and truncation. The first one generates a quantization noise with a smaller mean power than the latter, and it requires extra hardware to perform rounding. Truncation is usually preferred as an underflow technique in contrast to rounding, since the benefits obtained by rounding in terms of noise do not make up for the extra hardware needed. It is common practice to firstly estimate the signal dynamic range and set the scaling to a fixed value which avoids overflow, and then, perform word-length selection by assign-

Table 2.1: Summary of WLA approaches.

Approach	Scaling	Error est.	WLS	Algorithm	Cost	Comments
[BP00]	A: MIA	-	-	Acyclic LTI	-	Fixed-point MIA
[BR05]	S:	S:	O:EXS H:GRAD	Comm. systems	Area	System-level approach
[CCC ⁺ 06]*	A:TFA	A:TFA	O:ILP	LTI	Area	Resource sharing
[CCL03a]	A: TFA	A:TFA	O:ILP, H:GRAD	LTI	Area	High accuracy
[CFCNT04a]	S:	S:	H:	OFDM systems	Error	Opt. time reduced
[CH02,CH05]	H:	A:FBP	Not automated	Any	Area	User intervention required
[WKGM97]	A: IA	A:IA	Not automated	LTI	Error	User intervention required
[CLNT99]	A: MIA	-	-	Acyclic LTI	-	Produces range histograms
[Con03]	S:	H:	H:GRAD	Differentiable	Area	Underestimated variance
[CRS ⁺ 99]	H:IA	S:	H:	Any	Error	User intervention
[CSPL01, CSL02]	S:	S:	O:EXS, H:SA	Any	Area	Comparison of methods
[GCC06]	A:AA	A:AA	H:Adaptive SA	Acyclic diff.	Power	Approximation for NL operations
[GML04]	A: IA	A:Derivation	H:SA	Acyclic diff.	Area	Fixed-point and Floating-Point
[HE06]	S:	S:	O:EXS H:GRAD,SA	Any	Area	Error Comparison of methods
[HMS05]	S:	H:	H:GRAD	Any	Area	Resource sharing
[KKS95, KKS98, KS01]	S:	S:	H:GRAD	Any	Area	Scaling through statistical analysis
[LCCNT03a, LCCNT03b]	-	A:IA	-	LTI	Error	Several applications for IA
[LGC ⁺ 06]	A:AA	A:AA	H:ASA	Acyclic Diff.	Area	Approximation for NL operations
[Lóp04]	A:AA	A:AA	-	LTI	-	Application of AA to cyclic algorithms
[MRS ⁺ 01]	A:FBP	-	H:	-	Area	Error free User intervention
[MRSS04]	S:	H:	-	Acyclic diff.	-	Requires $ S ^{**}$ simulations
[MS02a]	A:TFA	A:TFA	H:NL opt.	LTI	-	High accuracy
[RB05]	A:FBP	S:	H:GRAD	Any	Error	Matlab-based
[SB04a, SB04b]	S:	H:	H:Quad. opt.	Differentiable	Area	Requires $ S ^2/2$ simulations**
[WP98]	A:IA	A:Derivation	-	Acyclic diff.	Area	Resource sharing
* in chapter 4			** $ S $ signals in algorithm			

ing the number of truncated LSBs that produces an output error smaller than a given noise constraint.

The allowed output noise of a circuit is normally measured in terms of the maximum peak-value of the output error or of the signal-to-quantization-noise ratio (SQNR) [OS87, CSPL01,KS01]. However, it can be also measured through other application specific metrics, such as the bit error rate (BER) in wireless systems [CFC⁺05,CFCNT04a,CCR01], etc.

Initially, the quantization of specific hardware systems was performed following the UWL approach [Liu71,OW72] inherited from the design of microprocessors. This approach enables a fast and simple quantization of an algorithm, but the results are considerably far from the optimum [KKS95,CCL03a]. When quantization is carried out assigning different word-lengths to each signal, then it is referred to as MWL quantization, and it has been extensively studied in the last few years [BR05,CFCNT04a,CFC⁺05,CVDM88,CCC⁺06,CLNT99,CRS⁺99,CCL03a,KKS95,KKS98,KS01,RB05,WP98]. The MWL approach calls for an adaption of the classic hardware design algorithms. The heterogeneity of word-lengths generates an explosion in the number of possible mappings between operations and variables to resources (e.g. functional units (FUs), storage resources and steering logic), thus requiring new ways to handle the design implementation. This issue is addressed in the next section.

A common classification of quantization methods for DSP algorithms distinguishes between analytical methods and simulation-based methods. Both types are applied to the quantization of MSBs (scaling) as well as to the quantization of LSBs (word-length selection). The analytical methods exploit certain analytical properties of the operations comprising the given algorithm. Although they produce results quickly, these suffer from overestimation, and they impose strong constraints to the kind of algorithms that they can work with (i.e. linear algorithms or algorithms with differentiable operations). The simulation-based methods use data vectors as inputs to the algorithm, and the produced outputs are used to analyze the quantization effect. On one hand, they tend to require exceedingly long simulation times

since the results highly depend on the selected input vectors if the number of data is small. On the other hand, their main advantage is that they can be applied to any type of algorithm.

The analytical methods to compute scaling include methods based on interval-arithmetic (IA) [BP00,CLNT99,GML04,WP98,WKGM97,CRS⁺99] and its extensions [Lóp04]; methods based on forward and backward propagation [MRS⁺01,RB05]; and methods based on the transfer function analysis [CCL03a,MS02a]. The methods based on IA, as well as those based on forward and backward propagation, tend to overestimate scaling [MRS⁺01,RB05,WKGM97,WP98], while approaches based on multi-intervals [BP00,CLNT99] and affine arithmetic (AA) [Lóp04] may overcome such overestimation and achieve good results. The methods based on AA and on the transfer function are suitable for feedback systems. All mentioned methods, except those based on the transfer function, are able to handle non-linear systems. The AA-based methods are modified when applied to non-linear systems to reduce the computational complexity, at the cost of some precision loss.

The simulation-based scaling methods [CSPL01,CSL02,KKS95,KKS98,KS01] use statistical properties of the signals, such as the mean and the variance, and the maximum and minimum peak values obtained during simulations, being suitable for both linear and non-linear systems. There are also hybrid methods which combine simulation-based and analytical techniques [CH02,CH05,CRS⁺99,WKGM97].

Once scaling has been computed for each signal, the general trend is to fix the number of MSB according to scaling and to start word-length selection to find the appropriate number of LSBs for each signal. However, there are some approaches that combine both scaling and word-length selection by means of saturation techniques [CCL03b]. Word-length selection requires, on one hand, a method to estimate the quantization effect on the algorithm precision, and on the other hand, a method to find a set of word-lengths which complies with the noise constraint while minimizing a particular design cost function. Needless to mention that both methods are closely related, since during the search of a solution it is necessary

to compute the quantization noise, so both methods are applied in conjunction. Once again, analytical and simulation-based methods are found in the literature.

The most important analytical methods are those based on the analysis of the transfer function [CCL03a, MS02a, MS02b], those based on the computation of the gradient between inputs and outputs [GML04, WP98] and those based on IA and its extensions [CH02, CH02, CH05, LGC⁺06, LCCNT03a, LCCNT03b, Lóp04, MRS⁺01]. These methods impose several restrictions such as their application only to linear systems with no feedback loops [LCCNT03a, LCCNT03b, MS02a, MS02b], systems with no feedback loops in general [CH02, CH05], linear systems with feedback loops [CCL03a, Lóp04] and differentiable systems [GML04, WP98]. The approach in [MRS⁺01] can be applied to linear and non-linear systems with feedback loops, but it requires the designer's intervention.

The simulation-based methods [CSPL01, CSL02, KKS95, KKS98, KS01] do not impose any restriction to the type of algorithm they cope with, and they allow the use of complex precision metrics. Their main drawback is that the estimation of the quantization noise effect is carried out via simulations, thus requiring long computation times. This is mainly due to the fact that simulation results vary with the input vectors used, and the only way to alleviate this, is to increase considerably the length of these. This situation makes word-length selection a time-consuming task. There are also hybrid techniques [Con03, CH02, CH05, CRS⁺99, WKGM97, MRSS04, SB04a, SB04b], being of special interest those that allow the generation of an approximate analytical expression of the quantization noise in non-linear systems by means of simulations [Con03, MRSS04, RMHS06, SB04a, SB04b], thus reducing the computation time of word-length selection considerably .

Word-length selection methods can be classified as optimal [BR05, CCC⁺06, CSPL01, CSL02, CCL03a, HE06] and heuristic [BR05, CFCNT04a, CCC⁺06, CSPL01, CSL02, CH02, CCL03a, GML04, KKS95, KKS98, KS01, GML04]. The optimal methods are mainly based on exhaustive search [BR05, CSL02, HE06] or in ILP techniques [CCL03a, CCC⁺06], both

requiring very long computation times. The heuristic methods are based on modifications of the gradient-descent algorithm [BR05, CCL03a, HE06, KKS95, KKS98, KS01] and on simulated annealing (SA) [GML04, CSPL01, CSL02, HE06, LGC⁺06] and they produce quasi-optimal results in standard computation times. The main advantage of ILP-based WLA is that it allows the mathematical representation of the problem under study and, although it is only applicable to small examples, the results obtained act as a reference to assess the quality of heuristic algorithms.

Referring to the optimization of the design cost (i.e. area, latency, power, etc.), quantization is commonly applied without directly considering cost as the objective function, but only the noise constraint [CRS⁺99, WKGM97, CFCNT04a, CFCNT04b]. The rationale is that a reduction of word-lengths can be directly translated into a reduction of the design cost, which is true in global terms but does not guarantee optimality. Works that include design cost within the optimization process obtain results closer to the optimum. The area cost is considered in [GML04, CCL03a, KS01, SK95, Con03, SB04a, CCC⁺06, HMS05, CH02, CH05, WP98], although in most cases (all but [KS01, WP98, HMS05, CCC⁺06]) a dedicated resource architecture (i.e. each operation has a dedicated resource assigned) is assumed. In [GML04], area and latency constraints are considered. There are some approaches which expose the effect of quantization on power consumption [Con03, HES04, JCC07, JC07, CGC05, CGC06], and only a few which optimize power [GCC06].

It is also interesting to highlight that only in [CCC⁺06, WP98, KS01, HMS05] the architecture of the implementation is not supposed to be fully parallel, and therefore a better optimization is performed as the interdependencies between the quantization noise and the datapath architecture are considered. The work in [CCC⁺06] is in fact developed in this thesis (see chapter 5).

Finally, the HLS methods in [CLCNT06a, CLCNT06b] considers presence of both LUT-based resources and embedded resources. These are some of the few works that tune HLS

to modern FPGAs. This issues are extended further in section 2.4. In summary, architectural information is normally neglected or constrained to fully parallel architectures during WLA, thus limiting the design space exploration. The benefits obtained by means of an MWL quantization are satisfactory when compared to an UWL approach, but the results are not guaranteed to be optimal. The inclusion of architectural information during WLA does improve the quality of implementations based on resource sharing [CCC⁺06, HMS05] and opens an interesting research line to be explored.

2.2. High-Level synthesis

High-Level synthesis (HLS) has had many interpretations since its definition in the early 90's [GDWL92, DM94, Lin97, MPC90, Cam90]. In this work, it is defined as the design task whose main goal is to transform the high-level description of an algorithm (i.e. a data flow graph (DFG)) into a Register-Transfer Logic (RTL) description, that complies with the design constraints. Among others, its phases are: *scheduling*, which aims at assigning *start* and *end* execution times to operations; *resource allocation*, which aims at the selection of the number and type of resources (functional units (FU), storage resources and steering logic) where operations are to be executed; and *resource binding*, whose goal is to assign specific resources to the operations and variables of the algorithm. Another important constituent of HLS is a set of component libraries corresponding to a given target technology (i.e. FPGA). These libraries contain models of the resources, as well as their RTL descriptions or netlists, and it is necessary to establish the compatibility between algorithm operations and library resources by means of the so-called *mapping*. Resource modeling must include architectural models (i.e. area), timing specifications (latency, data input/output sequence, etc.) and power consumption models. A proper component modeling and the correct definition of the operation-resource compatibility are key factors to produce high-quality system implemen-

Table 2.2: Summary of HLS approaches.

Approach	MWL	Combined HLS tasks	FU	Comments
[CCC ⁺ 06]*	✓	✓	Area and latency WL-wise	Latency clusters
[CLCNT06a, CLCNT06b]**	✓	✓	Area and latency WL-wise	Variable latencies Heterogeneous FPGAs
[CCL00a]	✓	✓	Area and latency WL-wise	Only resource binding
[CCL05]	✓	✓	Area and latency WL-wise	Variable latencies
[CFH ⁺ 05]	✓	×	Area and latency WL-wise	Latency clusters
[HMS05]	✓	×	Area WL-wise	Operation clusters
[KS01]	✓	×	Area and latency WL-wise	Operation clusters Latency clusters
[MRS ⁺ 01]	✓	×	Area and latency WL-wise	Operation clusters
[MRSMH06]	✓	×	Area and latency WL-wise	FU fragmentation
[WP98]	✓	×	Area and latency WL-wise	Operation clusters
[Ach93]	×	✓	Variable Latency Pipeline	Optimal
[LMD94]	×	✓	Complex FUs	Optimal
[CWP95, WP92, WP95]	×	✓	Bit/digit serial Digit parallel	Pipelined scheduling
* See section 5.1				
** Extended version in chapter 3				

tations.

The three main phases comprising HLS are commonly performed sequentially due to the complexity of the problem, scheduling being the first, followed by resource allocation and resource binding.

Before going any further let us introduce Table 2.2 which contains a summary of the main features of the HLS approaches that are going to be reviewed in this subsection. It must be stressed that this section is mainly focused on works oriented to MWL or that can be used as a reference to generate new MWL HLS algorithms.

The most popular scheduling techniques are those based on the "as soon as possible" (ASAP) and "as late as possible" (ALAP) algorithms [DM94], on lists [DM94], on the force-

directed algorithm [PK89] and on ILP [GN72]. The first three are heuristic algorithms with increasing computational complexity. The ILP-based techniques provide a compact mathematical description of scheduling and can be used as a reference to assess the quality of heuristic algorithms.

Resource allocation and resource binding are usually performed after scheduling and, in general, they are very simplified in practice; for instance, they are based on component libraries with a constant latency per resource type, or they use a fixed pipeline level, etc. It is also common to apply an UWL approach, thus further simplifying the set of available resources and the operation-resource compatibility rules. The most important methods are those based on graph theory (using compatibility graphs, clique partitioning, etc.) and the heuristic methods derived from them (left-edge algorithm, etc.) [DM94]. The works in [CCC⁺06, CLCNT06a, CLCNT06b, CCL00a, CCL05, HMS05, KS01, MRS⁺01, MRSMH06, WP98] are oriented to MWL systems and they achieve better hardware usage than traditional approaches (i.e. UWL). These works extend the model of operation-resource compatibility, since this was traditionally steered by the type of FU where the operations could be executed regardless of the word-length of both operation and FU. The new compatibility model considers the fact that an operation can be executed on an FU *iff* its input word-lengths are equal or smaller than the FU word-lengths. Area reductions of more than 50% are reported.

There are also proposals where several HLS phases are combined in order to perform a more complete design space exploration [Ach93, CCL00a, CCL05, CCC⁺06, CLCNT06a, CLCNT06b, LMD94, CWP95, WP92, WP95]. Most of these works assume uniform word-lengths. In [Ach93], HLS allows FUs with different latencies and pipeline levels, as well as multifunction FUs. In [LMD94], the previous work is extended to support complex FUs (i.e. MAC units), simultaneous execution of several additions using a single adder, and operation chaining. In [CWP95, WP92, WP95], it is possible to use heterogeneous FUs (e.g. bit-parallel, bit-serial and digit-serial), as well as to address the *iteration bound* [Par99] imposed

by feedback loops. These techniques can be used to face the mapping between operations and new embedded specialized FPGA resources. In [CCC⁺06, CLCNT06a, CLCNT06b, CCL05, HMS05] the phases of scheduling, resource allocation and resource binding are performed in a intertwined manner to address the synthesis of MWL algorithms.

Regarding design costs, HLS initially included area and latency constraints only. The resource-constrained synthesis aimed at minimizing latency, while complying with an area constraint, whereas the time-constrained synthesis focused on minimizing area, while complying with a latency constraint. Power consumption was introduced recently due to the importance of low-power design [Lin97, BM00].

Some important works where word-length information has been included within HLS and which are specially tuned for MWL systems, are [CCC⁺06, CLCNT06a, CLCNT06b, CJH96, CFH⁺05, CCL00a, CCL05, HMS05, KS01, MRS⁺01, MRSMH06, WP98]. In all these works, an operation can be executed only in FUs with a compatible word-length, that is, a word-length greater than or equal to the word-length of the operation. The cost of FUs is now dependent on the word-lengths, but some works tend to simplify some aspects of their modeling (grouping FU latencies [KS01, CCC⁺06, CFH⁺05], or grouping similar operations [HMS05, KS01, MRS⁺01, WP98]). In [CLCNT06a, CLCNT06b, CCL05] FUs are modeled with variable latencies which are word-length dependent. In [MRSMH06] FUs can be combined in order to minimize area waste due to the word-length difference between operations and FUs, while in [CJH96, KS01, MRS⁺01, WP98] this area waste is minimized by means of grouping operations with similar word-lengths. Only in [CCC⁺06, HMS05, KS01, WP98] WLA is integrated within HLS.

In summary, there are not many research works on HLS which consider word-length information, and the ones which actually integrate WLA within the synthesis process are indeed hard to find [CCC⁺06, HMS05, KS01, WP98]. This clearly outlines the fact that there are still many open research lines related to MWL HLS and to the combination of WLA and

Table 2.3: Summary of combined WLA and HLS approaches.

Approach	Algorithm	Tasks	Results	Optimization	Comments
[WP98]	LTI/NL	WLA (HLS estimates) HLS	UWL vs. MWL	Heuristic	1-cycle FUs No feedbacks
[KS01]	LTI/NL	WLA (minimum WLs) HLS WLA	No comparison	Heuristic	≈Variable latencies
[HMS05]	LTI/NL	loop: Clustering HLS WLA	UWL vs. MWL	Heuristic	Var. FU latencies
[CCC ⁺ 06]*	LTI	WLA+HLS	SEQ vs COMB	MILP	Optimal analysis 1-cycle FUs
* See section 5.1					

HLS. For instance, the impact of MWL on the steering logic and on the register allocation and binding phases are still to be researched. It is of special interest the analysis of new resource allocation and binding methods oriented to MWL systems. Also, the integration of scheduling and resource allocation for MWL systems is also a worthwhile line to follow.

2.3. Combined word-length allocation and high-level synthesis

There is very little research on the actual combination of WLA and HLS. In fact, works where both the architecture and the precision of the systems are shaped simultaneously, are really scarce. The combination of these two tasks provides a more complete design space exploration, and, therefore, improved design cost reductions are expected. Table 2.3 summarizes the information presented in this subsection. In this table *NL* stands for non-linear, *SEQ* for sequential approach and *COMB* for combined approach.

One of the pioneer works in this field is [WP98], where the word-length selection is

carried out by minimizing a lower bound on the area cost of a resource sharing architecture. Once the word-lengths are selected, scheduling, resource allocation and resource binding are performed. The interesting point of this work is that quantization makes use of an estimation of the final architecture of the system which is based on the algorithm latency and the signals word-lengths, instead of assuming a non-sharing implementation. The latency of resources is considered variable, although a very simple model is used. The results are compared to an UWL approach with significant area savings, but there is no comparison with a more traditional approach where the quantization is performed with no architecture information at all, or assuming a non-sharing system architecture.

In [KS01] WLA and HLS are interleaved. First, an initial quantization of the system is performed in such a way that the noise constraint is not met, although the quantization noise is quite close to the noise specification. With the obtained word-length information, a datapath is created by applying scheduling, resource allocation and resource binding. The resulting architecture is refined by means of increasing the FUs' word-lengths until the noise constraint is met. The work is valuable and has contributed to the field. However, the approach is too simplistic, since there is no room for further improvements because the interdependency between WLA and HLS is only exploited partially. Another issue that limits the scope of this work is the fact that FUs are supposed to have 1-cycle latencies, which is not a realistic assumption. Finally, the authors do not provide any comparison with a traditional two-step WLA and HLS procedure, so the validity of the approach remains uncertain.

The work in [HMS05] is an extension of [KS01] and [WP98]. Here FUs' costs (area, latency and power consumption) are word-length dependent. The paper presents some preliminary results of a combined WLA and HLS iterative approach. Basically, WLA and HLS are introduced within a loop. Operations are grouped after HLS based on mobility and word-length information. The word-length of each group is optimized by means of WLA in order to reduce datapath cost. Every change in the word-lengths of operations produces a new dat-

apath which leads to new groups of operations. The method iterates until there is no further improvement in the datapath costs. The idea seems interesting, but the paper fails to provide a significant set of benchmarks and results. Only one algorithm (IIR filter) is implemented and only one iteration is applied.

In [CCC⁺06] the possibilities of the combination of the two techniques are explored by means of an optimal approach based on MILP (see chapter 5 for details). Due to the extremely long computation times required by MILP the problem complexity is reduced by assuming 1-cycle latencies FUs and not considering storage or steering logic. The purpose of this work is to present initial results on the combined problem that can be used, as it will be seen throughout this text, as a starting point to face a more thorough approach. FUs' area reductions of up to 13% are reported.

There is a brief mention of the problem in [LGC⁺06], where resource sharing is known before quantization so it is possible to quantize including the actual area cost of the design. However, the treatment is too simplistic and it does not exploit the possibilities that offers the combination of WLA and HLS.

It is also worth mentioning [BCC05], where quantization and resource binding are performed in an interleaved manner for resource-dedicated datapaths. Here, the operations which require the greatest word-lengths are quantized and are assigned to resources first. In fact, they are assigned embedded FPGA multipliers, which are suitable for high word-length operations. By doing that, it is possible to implement operations with small word-lengths in LUT-based resources, therefore making a better use of the overall FPGA resources. The method produces good results, but the optimization criterium is based on minimizing the number of LUT-based resources and not on minimizing the overall resource usage.

There are still many open issues regarding the combination of WLA and HLS. The initial results on the actual combination of these two design tasks look promising, however there is still a lack of combined heuristics able to address industrial designs.

2.4. Field-programmable gate arrays

Last generation FPGAs [Alt, Xil] provide high integration densities (up to several tens of millions of equivalent logic gates), a high number of storage resources, specific resources for both data storage (RAM blocks) and digital signal processing (multipliers, configurable DSP blocks, etc.), and high clock speeds (up to 500 MHz). These features make them specially suitable for the implementation of complex DSP systems.

The creation of component libraries is essential to the correct application of HLS. Traditionally, this task has been carried out by synthesizing the RTL components (e.g. adders, multipliers, etc.) and using the resulting latencies and areas to characterize them [NHCB02, BFS04]. In some cases, a power consumption characterization has also been presented [BGP00]. The individual features of the components of a design (e.g. functional units, storage elements, steering logic, etc.) can be combined to obtain the design characteristics in terms of area, latency, etc. The final architecture latency is determined by the addition of the latencies of operations composing the critical path [NHCB02]. The term area is inherited from the ASIC technology, and it refers to the actual silicon area required for each component. In FPGAs this term is kept, but it is in fact measured as the number of resources (LUTS or flip-flops (FFs)) required for each component or for a given design. Having said that, the total area of a design is the addition of the areas of all its individual components [NHCB02]. Mean power consumption of each FU can be used combined with the switching activity factor of all their input signals to obtain a rough high-level estimation of power consumption [BGP00]. Recent works [CGC05, CGC06, JCC07, JC07] provide more accurate power estimates for FPGA arithmetic cores based on the switching activity and statistical properties of the FUs input signals.

Focusing on the definition of cost in terms of area, the emergence of specialized embedded FPGA resources makes possible that the same operation can be implemented using

different types of FPGA resources (i.e. traditional LUT-based resources vs. embedded resources), and even by their combination. Thus, area cannot be computed as the addition of the area related to the different resources, since they are of different kinds. For instance, a multiplication can be implemented by using LUT-based resources (i.e. Xilinx Virtex-II slices) or by using a dedicated embedded multiplier (i.e. Xilinx Virtex-II multBLOCK), and there is no equivalent to compare the area required for both implementations. This rises up the need for new mechanisms to compute area that take into account the existence of heterogeneous resources. In [BCC05] this problem is addressed and a new area metric is presented. The works in [CLCNT06a,CLCNT06b] integrate this metric within an automatic synthesis framework and present highly optimized MWL DSP implementations that exploits the benefits of both LUT-based and embedded FPGA resources (see chapter 3).

It is also important to stress the fact that due to the intrinsic features of FPGA architectures, most of the existent design algorithms are not directly applicable to them. For instance, it is not possible to reuse FUs by means of shared buses and it is necessary to utilize multiplexers, which are not efficiently implemented in current FPGA architectures, so their proper modeling and inclusion within the optimization process is essential. Works like [CC94,MN05] address this problem, although limited to UWL approaches. It is also necessary to analyze the capabilities that FPGAs offer to perform resource sharing, signal routing, etc., in order to be able to correctly apply HLS.

In the same way, the emergence of highly specialized blocks (i.e. DSP blocks) that are able to perform complex operations (e.g. FIR filtering, complex multiplications, etc.), demands new HLS techniques to cope with such complexity. The existing automatic approaches deal with the replacement of LUT-based resources for embedded multipliers [BCC05,CLCNT06a,CLCNT06b], for embedded memories [AO04,MCC05,Wil02], or for both [LVSB05]. These techniques result in important reductions in design cost, as well as in design time. Quantization is also affected by such specialized blocks, since the input and

output signal word-lengths are fixed a priori. Initially, it seems that this simplifies WLA, since these signals need not be quantized, but that proves untrue, since they come from, or are connected to, other blocks which might not have predefined signal word-lengths and, therefore, are subject to optimization.

Again, there are many open issues related to FPGA implementations of DSP algorithms. Current HLS approaches are not fully adapted to the recent advances in FPGA architecture. In most of the cases, the design methods are still based on the ASIC paradigm and it is necessary a migration to the reconfigurable computing paradigm, in order to make the most of FPGA implementations. The creation of new architecture models would allow optimizations levels far superior than the ones obtained up to date. These models must account for issues such as the existence of embedded specialized resources, data multiplexation, data storage, etc.

2.5. Conclusions

As a general statement, the MWL paradigm has been actively investigated in the past decade proving to be a very complex research topic, and there are still many interesting open issues in the field of DSP hardware design.

WLA achieves significant cost reduction in contrast to the UWL approach. However, most techniques do not account for architectural issues or they only address fully parallel implementations. Thus, the implementation results are not optimal if the final circuit architecture relies on resource sharing. Moreover, modern FPGA architectures are not considered and the benefits of the recently added DSP blocks are not fully exploited. Another important issue is that of computation time. Simulations are normally used to perform WLA leading to exceedingly large computation times. The existent works on fast noise estimators applicable to LTI and some non-linear systems must be improved further to allow fast WLA of complex

systems (i.e. communication systems).

As for HLS techniques, these are applied after WLA and are mainly targeting UWL systems. This again leads to non-optimal implementations. There are a few MWL HLS proposals, but there is still much work to do in this line. For instance, overlapped scheduled is not considered, which leads to highly optimized DSP implementations. Also, the inclusion of DSP-oriented FPGA architectures within MWL HLS is still a pending subject. The impact of multiplexers and specialized DSP and data storage blocks can be high within the MWL paradigm, but it has not been fully analyzed yet.

The combination of WLA and HLS is not a trivial task, considering that, as stated in the previous paragraph, these two tasks have still many open issues. The few existent works on the combined approach indicates that interesting benefits can be obtained. However, they have been developed in very simplistic terms. This indicates that the combined problem must be reformulated to address more realistic situations.

CHAPTER 3

HIGH-LEVEL SYNTHESIS OF DSP ALGORITHMS USING FPGAs

This chapter addresses the high-level synthesis of fixed-point DSP algorithms using programmable devices (FPGAs). The HLS of circuits has many interpretations. In this work it is referred to as the process that converts a mathematical description of an algorithm, along with some design constraints, into an RTL description which complies with the given restrictions.

The pursued synthesis method has as a main goal to produce highly optimized DSP implementations by means of adopting an MWL approach, as well as considering modern FPGA architecture issues, such as the use of embedded multipliers. On one hand, an MWL approach is selected since it has proved to provide significant cost savings (i.e. area, power, latency) compared to the traditional UWL approach. However, the benefits come with an extra computational cost, as the design space is increased considerably. On the other hand, the inclusion of embedded multipliers in the architecture model might produce cost reductions

but it requires the creation of new ways to measure cost in heterogeneous FPGA architectures (those composed of LUT-based and embedded resources), which are not necessarily compatible with existing cost estimators.

The chapter is divided as follows. Section 3.1 introduces high-level synthesis. Next, in section 3.2, the resource usage models are presented, as well as metrics to measure overall resource usage. Section 3.3 presents the optimization procedure used to implement HLS. Results are shown in section 3.4, where, first, the impact of using a UWL or MWL synthesis for both homogeneous and heterogeneous architectures is analyzed, and second, the impact of using embedded resources in MWL HLS is analyzed. Finally, in section 3.5, the conclusions are drawn.

3.1. High-level synthesis

This work focuses on the time constrained resource minimization problem [DM94]. The notation used is based on [DM94], and it is similar to that in [CCC⁺06,CLCNT06a,CLCNT06b,CCL05].

Given a signal flow graph $G_{SFG}(V, S)$, a maximum latency λ and a set of resources R (e.g. functional units R_{FU} , registers R_{REG} and steering logic R_{MUX}), it is the goal of the HLS procedure to find the time step when each operation is executed (*scheduling*), the types and number of resources forming R (*resource allocation*), and the binding between operations and variables to functional units and registers (*resource binding*) that comply with the constraints, while minimizing cost (i.e. area).

$G_{SFG}(V, S)$ is a formal representation of the algorithm, where V is a set of graph nodes representing operations, and $S \subset V \times V$ is a set of directed edges representing signals that determine the data flow. The operation set $V = V_F \cup V_M \cup V_G \cup V_A \cup V_D \cup V_I \cup V_O$ is composed of forks (signal branching), multiplications, gains (multiplications by a constant

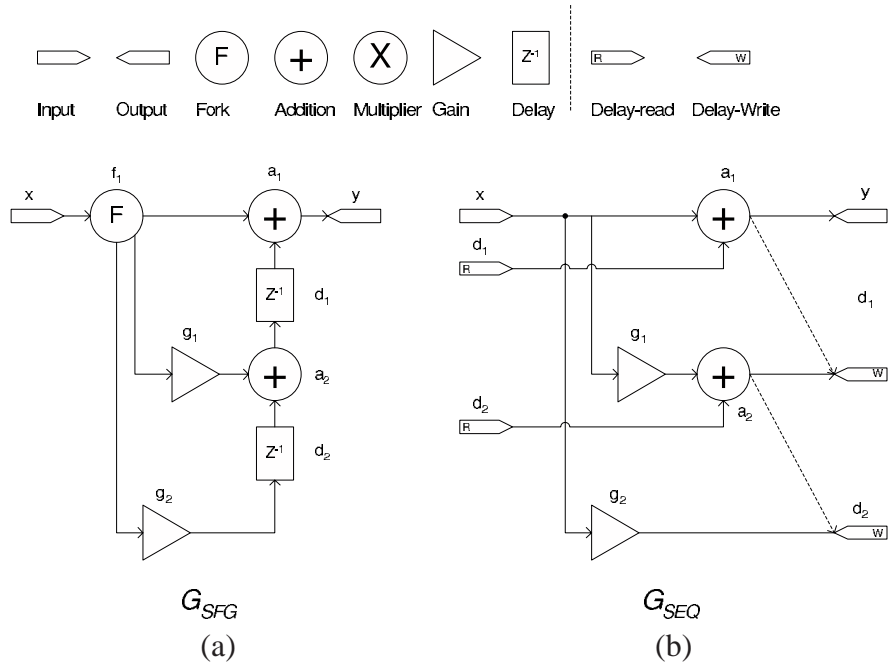


Figure 3.1: Conversion from G_{SFG} (a) into G_{SEQ} (b)

number), additions, unit delays, and input and output nodes. See Fig. 3.1-a for an example of a 2nd-order FIR filter. The set of incoming and outgoing edges of a node o is represented as $in(o)$ and $out(o)$. Signals are in two's complement fixed-point format, defined by the pair (n, p) , where n is the word-length of the signal not including the sign bit, and p is the scaling of the signal that represents the displacement of the binary point from the sign bit [CCL03a, CCC⁺06] (see Fig. 3.2). The values of the couples (n, p) have been computed during a previously performed WLA.

As aforementioned, the resources are divided into three types and they are used to form the so-called datapath (see Fig. 3.3). Functional units (R_{FU}) are in charge of executing the set of operations from V . Registers (R_{REG}) store the data produced by FUs and some intermediate values. Finally, steering logic (R_{MUX}) interconnects FUs and registers by means of multiplexers. The aim of HLS is to produce a minimum cost datapath that implements the given DSP algorithm using the resources from R .

The sequential graph $G_{SEQ}(O, E)$ is a representation of a single iteration of the algo-

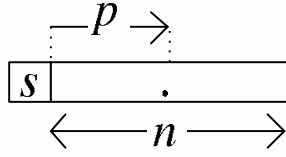


Figure 3.2: Fixed-point format

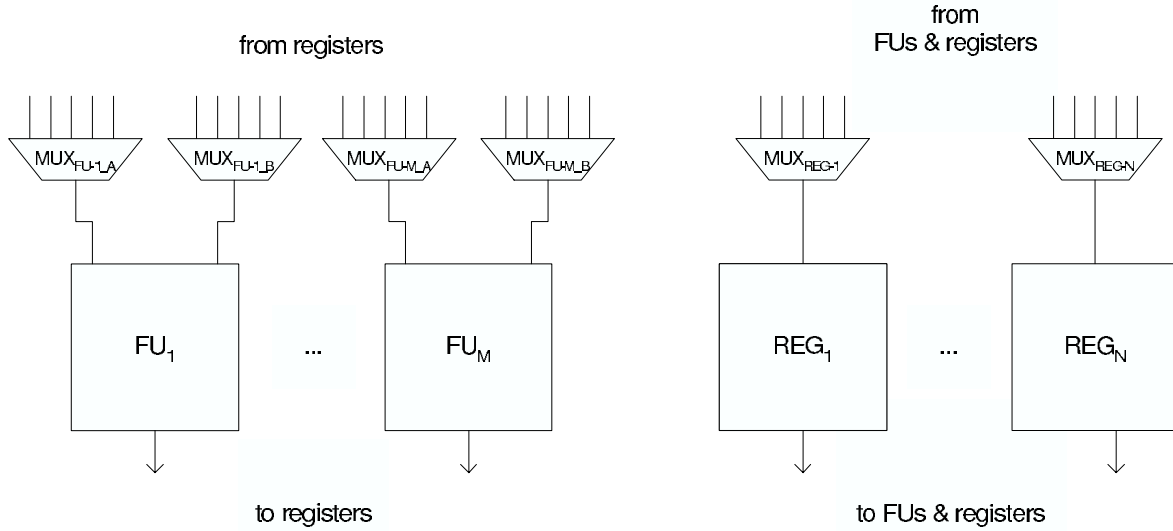


Figure 3.3: Datapath architecture: FUs, registers and multiplexers

rithm, which is used during the scheduling of operations (see Fig. 3.1-b). It is extracted from $G_{SFG}(V, S)$. Now the set of operations $O = O_M \cup O_G \cup O_A \cup O_{DR} \cup O_{DW} \cup O_I \cup O_O$ is composed of multiplications, gains, addition, delay readings, delay writings, and input and outputs. Any delay $v_D \in V_D$ is split into writing (O_{DW}) and reading (O_{DR}) nodes, acting the former as inputs and the latter as outputs of the sequential graph. Function $\theta(v) : V \rightarrow O$ indicates the relationship between the nodes from G and G_{SEQ} , which is straightforward (i.e. $\theta(V_A) = O_A$) except for delays where it is defined as $\theta(V_D) = O_{DR}$. There is also an inverse function $\theta^{-1}(o) : O \rightarrow V$, with the particularity that $\theta^{-1}(O_{DR}) = \theta^{-1}(O_{DW}) = V_D$. Signals

from S have been replaced by the set of directed edges E with no quantization information. These edges indicate data dependencies. Additional edges have been introduced to assure that the writing of a delay (o_{DW}) never occurs before all operations from O which operate the data from v_D have finished processing (see dash lines in Fig. 3.1-b). Also, forks have been removed since they are only necessary to compute the quantization noise at the output (see chapter 4).

$G_{SFG}(V, S)$ is also used to generate the available set of functional units R_{FU} . The set of functional units $R_{FU} = R_{ALUT} \cup R_{MLUT} \cup R_{MEMB}$ is composed of LUT-based adders, LUT-based generic multipliers, and embedded multipliers. This set of FUs covers FPGA devices from Xilinx families Virtex-II, Virtex-II Pro and Spartan-3, and is a subset of the resources present in Virtex-4 devices, and also in Altera devices from families Cyclone-II, Stratix and Stratix-II. All these devices make up a representative set of modern FPGA devices. An FU $r \in R_{FU}$ is defined by its type $type(r) = \{Adder_{LUT}, Multiplier_{LUT}, Multiplier_{EMB}\}$ and by its size. An $(m + 1) \times (n + 1)$ -bit multiplier ($m \geq n$) has a size composed of the couple (m, n) and an $(n + 1)$ -bit adder has a size n . A multiplier of size (n_{r1}, n_{r2}) can execute a multiplication with input word-lengths n_{v1} and n_{v2} , if $n_{r1} \geq n_{v1}$ and $n_{r2} \geq n_{v2}$. Additions $v \in V_A$ have an associated core word-length $n_{core,v}$, that depends on the fixed-point format pairs (i.e. (n, p)) of its inputs and indicates the minimum number of bits necessary for an adder to compute the operation. The core word-length is further explained in the next subsection. An adder r with a size of n_r bits can execute any addition with core word-length $n_{core,v}$ if $n_r \geq n_{core,v}$. The complete set of available FUs, R_{FU} , can be extracted from $G_{SFG}(V, S)$ following this set of compatibility rules.

Scheduling is expressed by means of function $\varphi : O \rightarrow Z^+$, which assigns a start time to each operation. The end time of an operation o can be computed if the latency λ_r of the resource r bound to o is known .

Resource binding, is divided into *FU binding* and *register binding*. *FU binding* makes

use of the compatibility graph $G_{COMP}(O \cup R, C)$ [CCL05], which indicates the compatible resources for each $o \in O$ by means of the set of edges $C \subset O \times R$. As previously mentioned, the set of available FUs R_{FU} is extracted from G_{SFG} and C is generated by applying the compatibility rules. The binding between operations and resources is expressed by means of function $\beta : O \rightarrow R \times Z^+$, where $\beta(o) = \{r, i\}$ indicates that operation o is bound to the i -th instance of resource r . The compatibility rules impose that $(o, r) \subset C$.

Register binding links variables $d \in D$ to registers $r \in R_{REG}$ by means of function $\gamma : D \rightarrow R_{REG} \times Z^+$, where $\gamma(d) = \{r, i\}$ indicates that variable d is bound to the i -th instance of register r . The set of variables D is extracted from O considering that there is a variable assigned to the output of each operation from the subset $O_M \cup O_G \cup O_A$ and to each delay o_D connected to another delay. Registers have an associated size n_r that determines the maximum allowed word-length of the variables bound to them. Inputs are supposed to be available during the whole execution of the algorithm, so there is no need for a storage element. Also, outputs do not require a register either.

Note that both R_{FU} and R_{REG} are composed of different resources, meaning that there are not two resources with the same type and size. For instance, if the datapath requires two 8x8-bit LUT-based multipliers, there would be two instances of a single resource $r \in R_{MLUT}$, such that $type(r) = multiplier_{LUT}$ and $size(r) = (7, 7)$.

The steering logic consists of the multiplexers required in front of FUs and registers to send data to and from these two types of resources as seen in Fig. 3.3. R_{MUX} is determined by φ , β and γ , since φ determines when data is generated, β when data is used by FUs, and γ where data is stored.

3.2. Cost estimation

3.2.1. Resource usage metric

The recent appearance of specialized blocks in FPGAs [Xil, Alt] calls for new design methods to efficiently exploit their advantages. Previous area estimation methods – or more precisely, resource usage estimation methods – for FPGAs were only applicable to LUT-based implementations [NHCB02], where the FPGA resources were basically composed of logic blocks (Configurable Logic Blocks, CLBs, in Xilinx devices or Logic Elements, LEs, in Altera devices) formed by several LUTs, flip-flops and some extra logic. When an operation can be implemented using different types of resources (i.e. LUT-based elements or embedded resources), it is necessary to have a method to wisely trade-off these resources in order to optimize the total resource usage. In [BCC05] such a method is introduced for the case of FPGA implementations of image filters with no resource sharing, and a measure of the area of a design with heterogeneous resources is proposed. However, the resource usage metric is not part of the optimization procedure, so the final results are not optimal. In [CLCNT06a, CLCNT06b] the infinite norm (∞ -norm) presented in [BCC05] as a heterogeneous resource usage metric, is used as the objective function of an SA-based optimization procedure, leading to highly optimized implementations within the framework of resource-sharing architectures for DSP algorithms. The benefits of using this method produced up to 60% of resource usage reduction, according to the new metric, when compared to LUT-based systems.

Given an FPGA with M different types of resources R_i ($i = 0 \cdots M - 1$), each type with a maximum number of $|R_i|$ resources, the resource requirements of a particular design d can

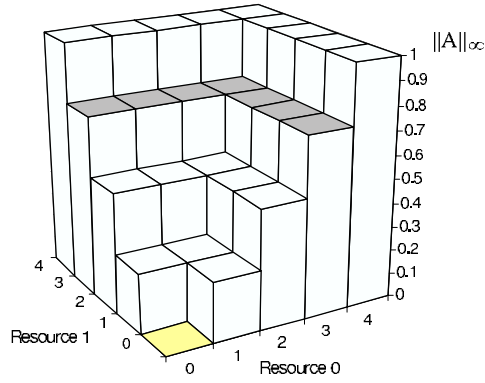


Figure 3.4: Normalized resource usage using ∞ -norm

be expressed as the following normalized area vector:

$$\hat{\mathbf{A}} \equiv \left\langle \frac{\#r_0}{|R_0|}, \frac{\#r_1}{|R_1|}, \dots, \frac{\#r_{M-1}}{|R_{M-1}|} \right\rangle \quad (3.1)$$

where $\#r_i$ is the number of resources of type R_i used. In [BCC05] is stated that the ∞ -norm of $\hat{\mathbf{A}}$ (eqn. 3.2) gives a measure of the area consumption of a particular design.

$$\|\hat{\mathbf{A}}\|_{\infty} = \max \left\{ \frac{\#r_0}{|R_0|}, \frac{\#r_1}{|R_1|}, \dots, \frac{\#r_{M-1}}{|R_{M-1}|} \right\} \quad (3.2)$$

More in detail, its inverse represents the number of times that the same implementation of design d can be replicated within the FPGA device. Fig. 3.4 displays the normalized values of ∞ -norm for a hypothetical FPGA with two types of resources R_0 and R_1 , with a maximum number of 4 resources of each type. The shaded region includes a set of resource distributions which produce the same ∞ -norm value, meaning that designs using any of these resource distributions can be replicated the same number of times. Any increase in the resource usage metric implies that the designs can be replicated a smaller number of times. The opposite happens for a decrease in the resource usage metric. Note that using ∞ -norm as a measure of resource usage, always leads to an implementation that, if replicated, achieves

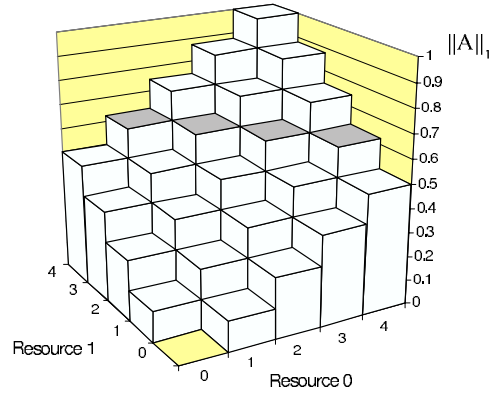


Figure 3.5: Normalized resource usage using 1-norm

maximum resource usage. However, this metric is not suitable when implementing several design blocks within an FPGA (or the same design block using different implementations). The reason for that is that design blocks with equal areas (∞ -norm) may differ in the total number of resources that they require. For instance, let us have two implementations a and b of the same design with area vectors $\mathbf{A}_a = \langle 2, 2 \rangle$ and $\mathbf{A}_b = \langle 2, 1 \rangle$. Implementation a requires 2 resources of both types 0 and 1, while implementation b requires only 1 resource of type 1. Both implementations can be replicated the same number of times, in fact, two times in our hypothetical FPGA. Using ∞ -norm as an architecture guidance, both implementations could be chosen as optimal since they rank as equally appropriate. However, the number of remaining resources in the FPGA is different depending on the solution adopted, which might lead to resource waste if more design blocks are to be implemented on the same FPGA. It would be necessary to introduce some information about the resource usage of all resource types and not only about the most used resource type.

Another interesting vectorial norm is 1-norm, which is basically the summation of all vector components.

$$\|\hat{\mathbf{A}}\|_1 = \sum_{i=0}^{i=M-1} \frac{\#r_i}{|R_i|} \quad (3.3)$$

Figure 3.5 shows the normalized values of 1-norm. This area metric contains information

about the trade-off between the different vector components: any increase in the number of resources of one type implies a decrease in the number of resources of the other types, if the area is to be kept constant. The shaded area includes implementations with resource distributions that produce the same resource usage metric. Now, an increase in 1-norm implies that the overall resource usage is increased, and a decrease implies that the overall resource usage has decreased. However, there is no connection with the number of times that those implementations can be replicated. As an example, let us have two implementations c and d of the same design with area vectors $\mathbf{A}_c = \langle 1, 3 \rangle$ and $\mathbf{A}_d = \langle 2, 2 \rangle$. Both implementations obtain the same 1-norm value, however implementation c can be implemented only once, while implementation d can be replicated twice.

In the work presented here a combination of ∞ -norm and 1-norm, called +-norm (*plus-norm*) is proposed and applied. A metric $\|\cdot\|_+$ that exploits the benefits of both norms but none of the drawbacks should fulfill the following conditions:

$$\forall i, j : \|\hat{\mathbf{A}}_i\|_+ < \|\hat{\mathbf{A}}_j\|_+ \Rightarrow \left(\|\hat{\mathbf{A}}_i\|_\infty < \|\hat{\mathbf{A}}_j\|_\infty \right) \vee \left(\left(\|\hat{\mathbf{A}}_i\|_\infty = \|\hat{\mathbf{A}}_j\|_\infty \right) \wedge \left(\|\hat{\mathbf{A}}_i\|_1 < \|\hat{\mathbf{A}}_j\|_1 \right) \right) \quad (3.4)$$

This can be expressed by means of a combination of the two norms.

$$\|\hat{\mathbf{A}}\|_+ = K \cdot \|\hat{\mathbf{A}}\|_\infty + \|\hat{\mathbf{A}}\|_1 \quad (3.5)$$

A feasible solution for K can be found by trying to comply with eqn. 3.6 for areas \mathbf{A}_1 and \mathbf{A}_2 , such that \mathbf{A}_1 requires only one type of resource, $\|\hat{\mathbf{A}}_1\|_\infty > \|\hat{\mathbf{A}}_2\|_\infty$ and $\|\hat{\mathbf{A}}_2\|_1$ has the biggest value that $\|\hat{\mathbf{A}}_1\|_\infty$ allows.

$$\|\hat{\mathbf{A}}_1\|_+ > \|\hat{\mathbf{A}}_2\|_+ \quad (3.6)$$

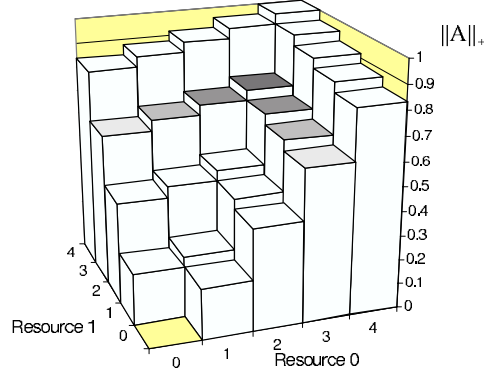


Figure 3.6: Normalized resource usage using $+$ -norm

First let us find upper bounds for $\|\hat{\mathbf{A}}_2\|_\infty$ and $\|\hat{\mathbf{A}}_2\|_1$:

$$\|\hat{\mathbf{A}}_2\|_\infty \leq \|\hat{\mathbf{A}}_1\|_\infty - \frac{1}{\max(|R_i|)} \quad (3.7)$$

$$\|\hat{\mathbf{A}}_2\|_1 \leq M \cdot \left(\|\hat{\mathbf{A}}_1\|_\infty - \frac{1}{\max(|R_i|)} \right) \quad (3.8)$$

Substituting 3.5, 3.7 and 3.8 into 3.6 allows:

$$K \cdot \|\hat{\mathbf{A}}_1\|_\infty + \|\hat{\mathbf{A}}_1\|_1 > (K + M) \cdot \left(\|\hat{\mathbf{A}}_1\|_\infty - \frac{1}{\max(|R_i|)} \right) \quad (3.9)$$

Since $\|\hat{\mathbf{A}}_1\|_1 = \|\hat{\mathbf{A}}_1\|_\infty$ and $\|\hat{\mathbf{A}}_1\|_\infty \leq 1$, a possible range of values of K that complies with 3.4 is expressed in terms of the number of types of resources (M) and the maximum number resources of any type ($\max(|R_i|)$).

$$K > (M - 1) \cdot (\max(|R_i|)) - M \quad (3.10)$$

Figure 3.6 shows the normalized values of $+$ -norm obtained as a function of resource usage. It is clear that this resource usage metric is the combination of ∞ -norm and 1-norm. In this case, if the overall resource usage (1-norm) is increased or decreased while keeping

constant the number of times that the design can be replicated (∞ -norm), there is a *small* increase or decrease in the $+$ -norm value. However, if the number of times the implementation can be replicated is increased or decreased, there is a *big* increase or decrease in the value of $+$ -norm. The shaded area contains designs that can be replicated the same number of times within the FPGA, but the different grey tones indicate the overall resource usage, where lighter tones implies that less resources are being used. In summary, $+$ -norm enables the selection of the resource distribution that allows replicating a design the maximum number of times within the FPGA, while using the minimum number of resources possible.

The conclusions presented here are also applicable to FPGAs with more than 2 types of resources ($M > 2$), and also when the number of total resources for each type differs ($\forall i, j : |R_i| < > |R_j|$). The metric $+$ -norm has a low computational cost and it is suitable for integer linear programming approaches [CCC⁺06].

3.2.2. Resource modeling

In subsection 3.1 resources were divided into three types: functional units (R_{FU}), registers (R_{REG}) and steering logic (R_{MUX}). In this subsection area and delay cost functions for these resources are presented.

Xilinx architecture

The cost models presented in this subsection are focused on Xilinx devices, but they can be easily adapted to other commercial devices. The Virtex-II architecture has been chosen since it forms the basis of modern FPGA architectures such as Spartan-3 and Virtex-4, and it is still a demanded prototyping device. The Virtex-II architecture includes the following blocks: Configurable Logic Blocks (CLB), RAM blocks, multiplier blocks and input/output blocks (IOB). This study focuses on CLBs and multiplier blocks. CLBs can be used to implement arithmetic cores, storage elements (registers) and steering logic (multiplexers).

Embedded multiplier blocks are used to implement 18x18 multiplications and can be extended to higher word-length multiplications by means of using several blocks plus some glue logic implemented on CLBs.

CLBs are made up of four sub-blocks called slices, which are selected in this work as the unit of area for the so-called LUT-based FPGA resources. Each slice is composed of two couples of LUTs and FFs and some glue logic. The FFs may store the data produced by its associated LUT. The area of a single LUT or a single FF is 0.25 of a slice and the glue logic is neglected. However, if a LUT is used and it is known that the FF is going to be wasted, a way to account for this situation is to increase the area of the LUT to 0.5 of a slice. The same applies to FFs whose associated LUT is not used at all.

Looking at Fig. 3.3 it can be inferred that registers can be implemented using the spare FFs from the multiplexers or FUs connected to them. Also, the multiplexers connected to FUs are clearly wasting FFs. These facts must be taken into account when estimating the area cost of the whole datapath.

Functional Units

The area of individual FUs is estimated as a function of the input and output word-length information (p and n), as in [CLCNT06a, CLCNT06b, CCL03a]. First, a sufficient number of resources (adders and multipliers) is implemented for different input word-lengths using a core generator (i.e. Coregen from Xilinx [Xil]). Then, the area and delay values reported by a synthesizer (i.e. Xilinx XST [Xil]) are used to obtain the area and delay functions (i.e. $A(r)$ and $D(r)$), fitted by means of a least squares approach. As a result, the costs are

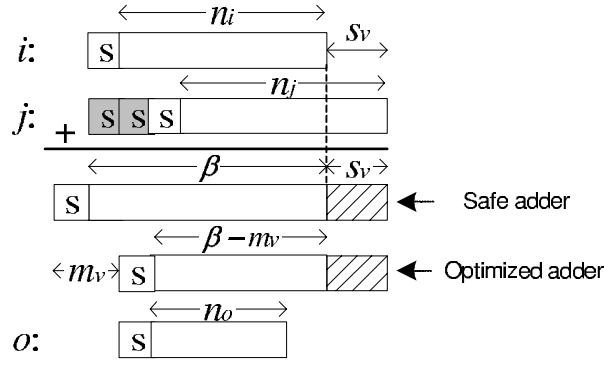


Figure 3.7: Computation of addition core word-length

modeled with expressions such as

$$\forall r \in R_{FU}$$

$$\mathbf{A}(r) = \langle a_3 \cdot n_1 + a_2 \cdot n_2 + a_1 \cdot n_1 \cdot n_2 + a_0, \quad (3.11)$$

$$b_3 \cdot n_1 + b_2 \cdot n_2 + b_1 \cdot n_1 \cdot n_2 + b_0 \rangle$$

$$D(r) = d_3 \cdot n_1 + d_2 \cdot n_2 + d_1 \cdot n_1 \cdot n_2 + d_0 \quad (3.12)$$

$$\lambda(r) = \lceil D(r)/T_{ck} \rceil \quad (3.13)$$

$\mathbf{A}(r)$ is the area vector, having as a first component the number of LUT-based resources and as a second the number of embedded multipliers. $D(r)$ is the delay in nanoseconds and $\lambda(r)$ is the delay in clock cycles, being T_{ck} the period of the system clock. The word-lengths of the resource inputs are n_1 and n_2 . The same expression can be used for 1-input resources by simply setting a_1, a_2, b_1, b_2, d_1 and d_2 to zero.

The area and delay of adders requires a special treatment. The implementation of an addition of two signals with the same (n, p) couple requires an $(n + 1)$ -bit adder (or an n -bit adder if there is no carry), which requires $n + 1$ LUTS, since each LUT implements a 1-bit full-adder. However, when the signals are not aligned and/or their word-lengths are different, the size of the adder is related to the number of bits that overlap between signals. The area

and delay can be expressed as functions of the core word-length of the addition bound to it: n_{core} . For a given addition $v \in V_A$ with inputs s_i and s_j and output s_o , the core word-length can be computed following equations 3.14-3.16. A ripple-carry adder is supposed since it is inherent to the FPGA architecture. Let us assume signals s_i and s_j comply with the following: signal s_j has its LSB positioned s_v bits to the right of the LSB of s_i , and scaling p_i should be bigger than or equal to the value of p_j (see Fig. 3.7 for an example). Equation 3.14 requires the definition of β and m . Let us define β as the number of overlapped bits between signals s_i and s_j with sign extension (eqn. 3.15). A *safe* adder would require $\beta+1$ bits to account for a possible overflow. Let us define m_v as the number of non-required bits at the output due to scaling (eqn. 3.16). Note that eqn. 3.16 makes use of the scaling of s_o , p_o . The size of an optimized adder is equal to the size of the safe adder minus m_v bits (eqn. 3.14).

$$v \in V_A,$$

$$n_{core,v} = \beta_v - m_v \quad (3.14)$$

$$\begin{aligned} \beta_v &= \max(n_j - s_v, n_i) + 1 \\ &= \begin{cases} n_j + p_j - p_i + 1, & \text{if } n_i - n_j + p_i - p_j \geq 0 \\ n_i + 1, & \text{otherwise} \end{cases} \end{aligned} \quad (3.15)$$

$$m_v = (\max(p_i, p_j) + 1) + p_o \quad (3.16)$$

Once n_{core} is computed, equations 3.11-3.13 can be used by substituting n_1 by n_{core} .

The area and delay costs of embedded multipliers are not word-length dependent. This work only considers 18x18 multipliers, which require one embedded block, and 36x18 multipliers which require two embedded blocks and some extra logic.

Table 3.1 shows the area and delay parameters obtained for resources of an XC2V40-6

Table 3.1: Area and delay parameters of FUs and registers.

Resource	a3	a2	a1	a0	b3	b2	b1	b0	d3	d2	d1	d0
Mult.	-0.55	-0.55	0.62	16.57	0.00	0.00	0.00	0.00	0.041	0.041	0.0089	5.37
Addition	0.50	0.00	0.00	0.5	0.00	0.00	0.00	0.00	0.042	0.00	0.00	5.89
Reg.	0.25	0.00	0.00	0.25	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
MULT_{18×18}	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00	0.00	0.00	0.00	5.39
MULT_{36×18}	0.00	0.00	0.00	19.0	0.00	0.00	0.00	2.00	0.00	0.00	0.00	10.47

device.

Registers

Registers (R_{REG}) are treated as 1-input resources with zero delay. Each bit of a register is stored in a FF, so the area model is straight forward (see table 3.1). Note that the area of a 1-bit register is a quarter of a slice, thus modeling that it is using a spare FF from a slice. From Fig. 3.3 it is clear that a register input is likely to come from a multiplexer or directly from an FU, hence the 0.25 values of a_3 and a_0 that indicate the possibility of using the spare LUTs in the slice.

Steering logic

The area of multiplexers in UWL systems is only affected by the data word-length, which sets the multiplexer size, and by the number of different data sources (e.g. registers or FUs), which determines the multiplexer width. An estimation of the area of an N -input multiplexer of word-length M for Virtex-II devices is given by

$$A_{MUX} = M \cdot N/4 \text{ slices} \quad (3.17)$$

In MWL systems, data must be aligned before being processed by FUs or stored by registers. In [SGDM93] the problem of data alignment and multiplexing is tackled. Multiplexers steer

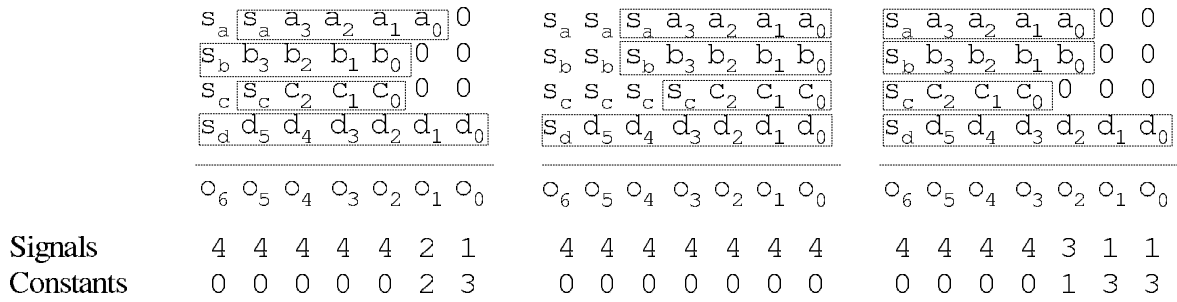


Figure 3.8: Different signal alignment approaches: a) arbitrary alignment, b) LSB alignment, c) MSB alignment

data from FUs to registers and vice versa, and some blocks are introduced before multiplexers to perform data alignment. The article exposes the complexity of the alignment problem, but its goal is not to minimize the area of interconnections but simply to find a feasible implementation that handles data alignment. Data is fit into the multiplexer size by means of positioning data with an *offset* with respect to the input bus LSB and then setting the remaining signals to '0', '1', the sign bit, or labeling the signals with the 'don't care' value.

In this work, the situation is simplified by means of considering only sign extension and zero padding. Also, multiplexers are used for both data multiplexing and data alignment, since the combination of these two tasks leads to a reduction in the number of control signals, and therefore, control logic. In addition, the chances for logic optimization are greater than if two separate blocks (an alignment block and a multiplexer) are used.

Let us first analyze the implications of multiplexing MWL signals. Fig. 3.8 presents three different types of alignments for a 4-input multiplexer. Fig. 3.8-a depicts an arbitrary alignment, Fig. 3.8-b an alignment to the right or LSB alignment, and Fig. 3.8-c represents an alignment to the left or MSB alignment. Note that sign extension (Fig. 3.8-a and Fig. 3.8-b) does not offer any opportunity for logic optimization, while zero padding (Fig. 3.8-a and Fig. 3.8-c) does offer it, due to the reduction in the number of signals and the introduction of constant bits (zeros) that can be hard-wired into the multiplexer logic. In fact, it is MSB alignment (Fig. 3.8-c) the option that allows a greater logic reduction. Therefore, it is

recommended to apply this alignment whenever possible.

Only adders impose requirements on the alignment of their inputs (see Fig. 3.7). If all signals with the larger scaling (p) are assigned to one of the input multiplexer (i.e. multiplexer connected to input a : MUX_a) and the other signals are assigned to the other input multiplexer (MUX_b), it is possible to apply MSB alignment to the first multiplexer (MUX_a). The other multiplexer would have an alignment similar to that in Fig. 3.8-a, so the chances for optimization are smaller. Multipliers can have any alignment in their input signals, with the only effect that the output signal is shifted depending on the input alignment, and this fact can be handled by the register multiplexers. Therefore, an MSB alignment can be used in both of their multiplexers, thus reducing their area. Moreover, applying MSB alignment results in a concentration of the output's MSBs to the left, which is a desirable feature because it simplifies output alignment.

The multiplexers of registers can also benefit from MSB alignment. But in this case the optimization can be even greater because it is not necessary to perform any zero padding, as this is performed by the FU's multiplexers. MSB alignment comes with an extra benefit, since all signals coming from the same register and going to an MSB-aligned FU multiplexer are already aligned, no matter the word-length of the stored signal. This does not occur with LSB alignment.

Hundreds of combinations of data word-lengths, number of data sources and sorting of multiplexer's inputs, were tested on MSB-aligned multiplexers using different commercial tools (Xilinx XST 7.1 [Xil] and Synplicity Synplify Pro 8.4 [Syn]), in order to obtain an expression of the area cost of multiplexers. The implementation results changed greatly depending on the sorting of input data and with the tool used, which made difficult to extract a reliable and tool-independent model for the area cost. However, it was possible to extract a lower bound on the multiplexers' area. If the word-lengths are heterogeneous, being M the maximum word-length present, a lower bound on the area cost of multiplexers can be

computed as in

$$\underline{A}_{MUX} = \frac{1}{4} \sum_{i=0}^{N-1} \{(n_i + 1) + s_i\} \quad (3.18)$$

where n_i is the word-length of signal i and s_i is the number of sign bits extended. Note that if there is MSB alignment $s_i = 0$.

Delay of routing and steering logic

During HLS it is difficult to estimate the impact of routing delays in the performance of the system. A simple way to account for this is to multiply the clock period by a factor $\alpha_{T_{ck}}$ (smaller than 1.0) during HLS [EJCT00]. In a similar fashion, the delay of multiplexers can affect the system performance. Using a conservative factor $\alpha_{T_{ck}}$ (i.e. 0.7) can account for both routing and multiplexing delays.

3.3. Optimization procedure

The optimization procedure is based on Simulated Annealing [KGV83] and it is shown in algorithm 3.1. The inputs are the signal flow graph $G_{SFG}(V, S)$ and the total latency constraint λ . The optimization procedure determines the set of resources of the datapath $R = R_{FU} \cup R_{REG} \cup R_{MUX}$, the scheduling φ , the FU binding β , and the register binding γ , which define the datapath, the steering logic, and the timing of the control signals. First, the sequential graph $G_{SEQ}(O, E)$, the set of functional units R_{FU} , the set of registers R_{REG} , and the compatibility graph $G_{COMP}(O, R_{FU})$ are extracted (line 1). An initial resource mapping m_0 is selected by mapping operations to the fastest LUT-based compatible resources available (line 2), and the area A_0 occupied by the resulting datapath is used as the initial area to guide the SA-based optimizer (line 3). From this point (line 5-30), the optimization proceeds following the typical SA behavior: the algorithm iterates while producing changes (line 8) – also referred to as *movements* – that modify the value of the cost function

(i.e. area) until a certain exit condition is reached. If these changes lead to a cost reduction, they are accepted (line 11), if not, they are accepted with a certain probability which depends on the current temperature T (line 15). The temperature starts at a high value and decreases with time. The initial temperature allows an area increase of a 5% with a probability of 0.9. Most movements are accepted at the beginning of the process, thus enabling a wide design space exploration. As temperature decreases, only those movements which produce small cost deviations are accepted. The temperature is decreased when the *equilibrium state* is reached (line 19). Sporadic *restarting* [BMU92] is also allowed (line 27), which repositions the optimization variables at the last minimum state found.

The variation in cost ΔA is normalized with respect to the initial area A_0 (line 10). This is a simple way to control that the behavior of SA is not affected by the complexity of the algorithm [OKSH06], which it is approximated by $\left\| \hat{A}_0 \right\|_n$. The value of n must be set to 1 (or ∞) for homogeneous-architecture FPGAs, and to '+' for heterogeneous-architecture FPGAs.

The temperature is kept constant until either the *equilibrium* or the *frozen* state is reached [KGV83]. The temperature is decreased lineally using factor $\alpha_T = 0.95$ (see lines 19-26). The equilibrium state is reached when the number of accepted movements, stored in variable *accepted*, is greater than a certain threshold L which is defined in 3.19. The frozen state is reached when the number of iterations is greater than a certain threshold L' (eqn. 3.20) ($L' > L$). When the frozen state is reached for three consecutive times, then the optimization procedure stops.

$$L = |O| \times |O| \quad (3.19)$$

$$L' = k \cdot L, k > 1.0 \quad (3.20)$$

The *restart* process resets the current mapping and area to the best recorded so far

Algorithm 3.1 Optimization procedure

Input: $G_{SFG}(V, S), \lambda$ **Output:** $R = R_{FU} \cup R_{REG} \cup R_{MUX}, \varphi, \beta, \gamma$

- 1: Extract $G_{SEQ}(O, E), G_{COMP}(O, R_{FU}), R_{FU}, R_{REG}$
 - 2: Find initial mapping m_0
 - 3: Compute initial area \mathbf{A}_0 from m_0
 - 4: $\mathbf{A}_{min} = \mathbf{A}_{last} = \mathbf{A}_0$
 $m_{min} = m_{last} = m_0$
 $T = T_0$
 $iteration = accepted = exit = 0$
 - 5: **while** $\neg exit$ **condition do**
 - 6: $m = m_{last}$
 - 7: $iteration = iteration + 1$
 - 8: Perform change to current m
 - 9: Compute area \mathbf{A} from m (algorithm 3.2)
 - 10: $\Delta A = \frac{\|\hat{\mathbf{A}}_{last}\|_n - \|\hat{\mathbf{A}}\|_n}{\|\hat{\mathbf{A}}_0\|_n}$
 - 11: **if** $\Delta A < 0$ **then**
 - 12: $\hat{\mathbf{A}}_{min} = \hat{\mathbf{A}}_{last} = \mathbf{A}$
 $m_{min} = m_{last} = m$
 $accepted = accepted + 1$
 - 13: **else**
 - 14: $p = e^{\frac{\Delta A}{T}}, r = \text{rand}[0 \dots 1)$
 - 15: **if** $r \leq p$ **then**
 - 16: $\mathbf{A}_{last} = \mathbf{A}, m_{last} = m$
 $accepted = accepted + 1$
 - 17: **end if**
 - 18: **end if**
 - 19: **if equilibrium state then**
 - 20: $T = \alpha_T \cdot T$
 - 21: $iterations = accepted = exit = 0$
 - 22: **else if frozen state then**
 - 23: $T = \alpha_T \cdot T$
 - 24: $iterations = accepted = 0$
 - 25: $exit = exit + 1$
 - 26: **end if**
 - 27: **if restart condition then**
 - 28: $\mathbf{A}_{last} = \mathbf{A}_{min}$
 $m_{last} = m_{min}$
 - 29: **end if**
 - 30: **end while**
-

(line 27). The restart condition happens when the current area reaches a value higher than 150% of A_{min} . It is meant to avoid design space exploration of high cost regions.

The changes on the cost function (line 8) are performed by applying with equal probabilities the following movements to the resource mapping function m :

- M_A : Map an operation $o \in O$ to a non-mapped resource.
- M_B : Map an operation o to another already mapped resource.
- M_C : Swap the mapping of two compatible operations mapped to different resources.

The computation of the area cost is shown in algorithm 3.2. First, it is checked whether the current resource mapping m complies with latency λ (lines 1-4). If it does not, the actual latency λ' is computed. Later on (line 26), any deviation from the design constraints is penalized by means of increasing the area cost of the solution. Thus, solutions that do not meet the latency constraint are included within the design space exploration [Lue84, LV99].

Then, the resource allocation and resource binding that minimizes FU area is sought by means of a loop where several list-based scheduling operations are performed (lines 4–18). The purpose of the loop is to check different combinations on the number of instances of the resources. Both lower and upper bounds on the number of instances for each resource are computed (line 6). All combinations of possible instances are computed and stored in the set of vectors I . The list-based scheduling performs an ASAP scheduling to the operations sorted by mobility in ascendant order, providing a fast way to find a valid solution. Note that the size of I is being pruned while the loop iterates: all combinations of FU instances that require areas greater than the minimum found so far are removed (line 15). Thus, resource allocation is sped up.

Once the minimum FU area scheduling is found, the datapath is defined. The tasks of register binding and multiplexer allocation are not commonly included within the optimization loop, in spite of their impact in the final architecture. In this work, these two tasks are

Algorithm 3.2 Computation of area cost

Input: $G_{SEQ}(O, E), \lambda$ **Output:** $R = R_{FU} \cup R_{REG} \cup R_{MUX}, \varphi, \beta, \gamma, \mathbf{A}$

- 1: Compute minimum latency λ' for mapping m
 - 2: **if** $\lambda' < \lambda$ **then**
 - 3: $\lambda' = \lambda$
 - 4: **end if**
 - 5: Find set of functional units $R'_{FU} = \{r'_0, \dots, r'_{N-1}\}$ with mapped operations
 - 6: $\forall r' \in R'_{FU}$: Compute instances lower bound $\underline{inst}(r')$ [OKD97] and upper bound $\overline{inst}(r')$
 - 7: $I = \{\text{all vectors ranging from } \langle \underline{inst}(r_0), \dots, \underline{inst}(r_{N-1}) \rangle \text{ to } \langle \overline{inst}(r_0), \dots, \overline{inst}(r_{N-1}) \rangle\}$

 - 8: $\mathbf{A}_{FUmin} = \infty$
 - 9: **for** $i \in I$ **do**
 - 10: $\forall r'_j \in R'_{FU}, inst(r') = i[j]$
 - 11: $\mathbf{A}' = \sum_{r' \in R'_{FU}} \mathbf{A}(r') \cdot inst(r')$
 - 12: **if** $\|\hat{\mathbf{A}}'\|_n < \|\hat{\mathbf{A}}_{FUmin}\|_n$ **then**
 - 13: **if** list_scheduling(λ', m) **then**
 - 14: $\mathbf{A}_{FUmin} = \mathbf{A}'$
 - 14: $\hat{i}_{min} = i$
 - 14: $\varphi_{min} = \varphi$
 - 14: $\beta_{min} = \beta$
 - 15: $I = \{i \in I : \|\sum_{r' \in R'_{FU}} \hat{\mathbf{A}}(r') \cdot inst(r')\|_n < \|\hat{\mathbf{A}}_{FUmin}\|_n\}$
 - 16: **end if**
 - 17: **end if**
 - 18: **end for**
 - 19: $\forall r'_j \in R'_{FU}, inst(r') = \hat{i}_{min}[j]$
 - 20: $\varphi_{min} = \varphi$
 - 20: $\beta_{min} = \beta$
 - 21: Extract D, R_{REG}
 - 22: Compute register binding
 - 23: Extract R_{MUX}
 - 24: $R = R_{FU} \cup R_{REG} \cup R_{MUX}$
 - 25: $\mathbf{A}' = \sum_{r \in R} \mathbf{A}(r) \cdot inst(r)$
 - 26: $\alpha_\lambda = (\lambda' - \lambda) / \lambda$
 - 27: $\mathbf{A} = \mathbf{A}' \cdot (1 + \alpha_\lambda)$
 - 28: **if** $\|\hat{\mathbf{A}}\|_n < \|\hat{\mathbf{A}}_{min}\|_n$ **then**
 - 29: $\mathbf{A} = \mathbf{A}_{min} \cdot (1 + \alpha_\lambda)$
 - 30: **end if**
-

part of the optimization procedure. However, they are implemented using fast heuristics, which produce non-optimal results, for the sake of reducing optimization time. Since the FU allocation and binding is performed in a quasi-optimal way and the area that FUs occupy dominates the total area of the datapath, it seems sensible to simplify register binding and multiplexer allocation.

Register binding is performed by applying a *left-edge* algorithm [DM94]. Inputs signals are supposed to be available for all λ cycles and do not require storing. Each variable assigned to a delay is initially assigned a register, and after that, the left-edge algorithm is applied as usual.

From sets R_{FU} and R_{REG} and functions φ , β and γ it is possible to extract the steering logic resources R_{MUX} . Registers have a single multiplexer (see Fig. 3.3), while FUs have two. The FU multiplexers correspond to the multiplexers connected to the inputs a and b ($n_a \geq n_b$) of multipliers and adders. First, a list of all the operations bound to FUs, and a list of variables bound to registers are generated using the information in β and γ . The generation of register multiplexers is straight forward: there is a multiplexer if the register input data come from different sources (i.e. FUs and registers). All signals bound to the register are inputs to its multiplexer. An MSB alignment is applied to optimize area (see subsection 3.2.2). However, the generation of FU multiplexers is more complex. Let us start with adders. In subsection 3.2.2, it was concluded that a way to optimize the area of adders' multiplexers was, first, to arrange the additions inputs as in Fig. 3.7 ($p_i > p_j$) and then assign all signals i to the same multiplexer (MUX_a) and all signals j to the remaining multiplexer (MUX_b). MUX_a permits an MSB alignment, while MUX_b requires a custom alignment such in Fig. 3.8-a. Multipliers can perform generic multiplications and constant multiplications (i.e. gains) with input signals i and j . The former case is handled by assigning the multiplication input with the largest word-length to MUX_a and the other input to MUX_b , using an MSB alignment. The latter case implies that one of the inputs is a constant.

Assuming that all constants are stored in memory (i.e. a LUT-based distributed RAM), all constant signals must be treated as a single signal with a word-length equal to the maximum word-length of all constants. The process goes as explained before for generic multipliers. Finally, the control signals can be easily extracted from the scheduling contained in φ .

The area vector is computed by adding the area of each resource multiplied by the number of instances required (line 25). If $\lambda' > \lambda$ the area is penalized by means of factor α_λ . If the implementation does not comply with the latency constraint and if the resulting penalized area is smaller than A_{min} , then the area is forced to be bigger than A_{min} (see line 28).

Summarizing, the optimization procedure is actually controlled by changing iteratively the mapping between operations and FUs. These changes impact on the structure of the data-path and, therefore, on its area cost, which is the function to be minimized. This method provides a robust way to simultaneously perform the tasks of scheduling, resource allocation and resource binding for multiple word-length systems. A simplified version of the procedure exposed above was satisfactorily applied to MWL systems in [CLCNT06a, CLCNT06b].

3.4. Results

The results are divided in four sections:

- i) The effect of using an UWL approach in contrast to an MWL approach is exposed for homogeneous FPGA architectures. Even though it seems obvious that area reductions are expected, it is very important to quantize the effect of supporting an MWL approach, since most of the HLS algorithms are based on UWL premises. The analysis is aimed at justifying the necessity of implementing new MWL HLS algorithms instead of adopting the existent UWL-based ones.
- ii) A similar analysis applied to heterogeneous systems is presented. The use of high-precision embedded multipliers (18x18-bit multipliers blocks) may lead to thinking that

an UWL WLA suffices to obtain area efficient implementations. The comparison of UWL vs. MWL heterogeneous will reveal if there are benefits and if these benefits are high enough to justify the implementation of MWL HLS algorithms for heterogeneous architectures.

- iii) Results related to the implementation of MWL systems that include both LUT-based and embedded resources are presented in subsection 3.4.3. Again, though it is expected that the area cost is highly reduced by means of embedded resources, it is important to quantize this reduction, to see if it is worth it to developed efficient HLS techniques to trade-off LUT-based and embedded resources.
- iv) Subsection 3.4.4 deals with the effect of including the cost of registers and multiplexers in the cost function used in the HLS optimization loop. The inclusion of registers and multiplexers in the cost function clearly increases the complexity of the HLS algorithms, so it is necessary to analyze the benefits of this approach.

The following benchmarks are used for the analysis: (i) an ITU RGB to YCrCb converter (*ITU*) [CCL03a]; (ii) a 3rd-order lattice filter (*LAT₃*) [Par99]; (iii) a 4th-order IIR filter (*IIR₄*) [KKS00]; and (iv) an 8-th order linear-phase FIR filter (*FIR₈*). The number of operations and signals required for each benchmark is shown in table 3.2. All algorithms are assigned 8-bit inputs and 12-bit constant coefficients. The algorithm implementations have been tested under different latency and output noise constraint scenarios assuming a system clock of 125 MHz. In particular, the noise constraints were $\sigma^2 = \{10^{-k}, 10^{-(k+1)}, 10^{-(k+2)}\}$, where k is the minimum number that makes 10^{-k} as close as possible to the variance of the quantization noise that would present the output of the benchmark if quantized to 8 bits ($n = 7$).

The target devices belong to the Xilinx Virtex-II family. The area results are normalized with respect to the XC2V40 device (256 slices, 4 embedded 18x18 multipliers) and expressed according to Eqn. 3.2. For instance, an area vector with ∞ -norm equal to or smaller than 1.0

Table 3.2: Characteristics of benchmarks.

Alg.	IN	OUT	+	*	Forks	Delays	Nodes	+/*	Signals
<i>ITU</i>	3	3	4	6	4	0	20	10	21
<i>LAT₃</i>	1	1	8	9	9	3	31	17	36
<i>IIR₄</i>	1	1	8	7	6	4	27	15	34
<i>FIR₈</i>	1	1	8	5	8	8	31	13	38

implies that the device XC2V40 is the smallest-cost device able to hold the design, whereas a ∞ -norm greater than 1.0 and equal to or smaller than 2.0 implies that the smallest-cost device able to hold the design is the XC2V80, and so on.

The results are obtained by means of executing numerous SA optimization tasks. In fact, for each benchmark it is necessary to perform an SA optimization for each possible combination of latency and quantization noise variances considered in the analysis. Moreover, due to the random nature of SA, each optimization is repeated four times, and the implementation that offers the best cost is chosen, in order to reduce variance in the final results.

Before HLS, each algorithm is translated to a fixed-point specification by means of two word-length optimization procedures, that follow an UWL approach and an MWL approach, respectively. The UWL synthesis is achieved by means of computing the maximum scaling required to avoid overflow, and then computing the minimum word-length that if applied to all signals complies with the noise constraint; after that, all signals are homogenized by acquiring the same fixed-point format (i.e. $(max(p), n)$). The MWL synthesis is achieved by means of an SA-based approach, which minimized the area of a resource-dedicated implementation (with no resource sharing). This procedure is detailed in the next chapter (see section 4.3).

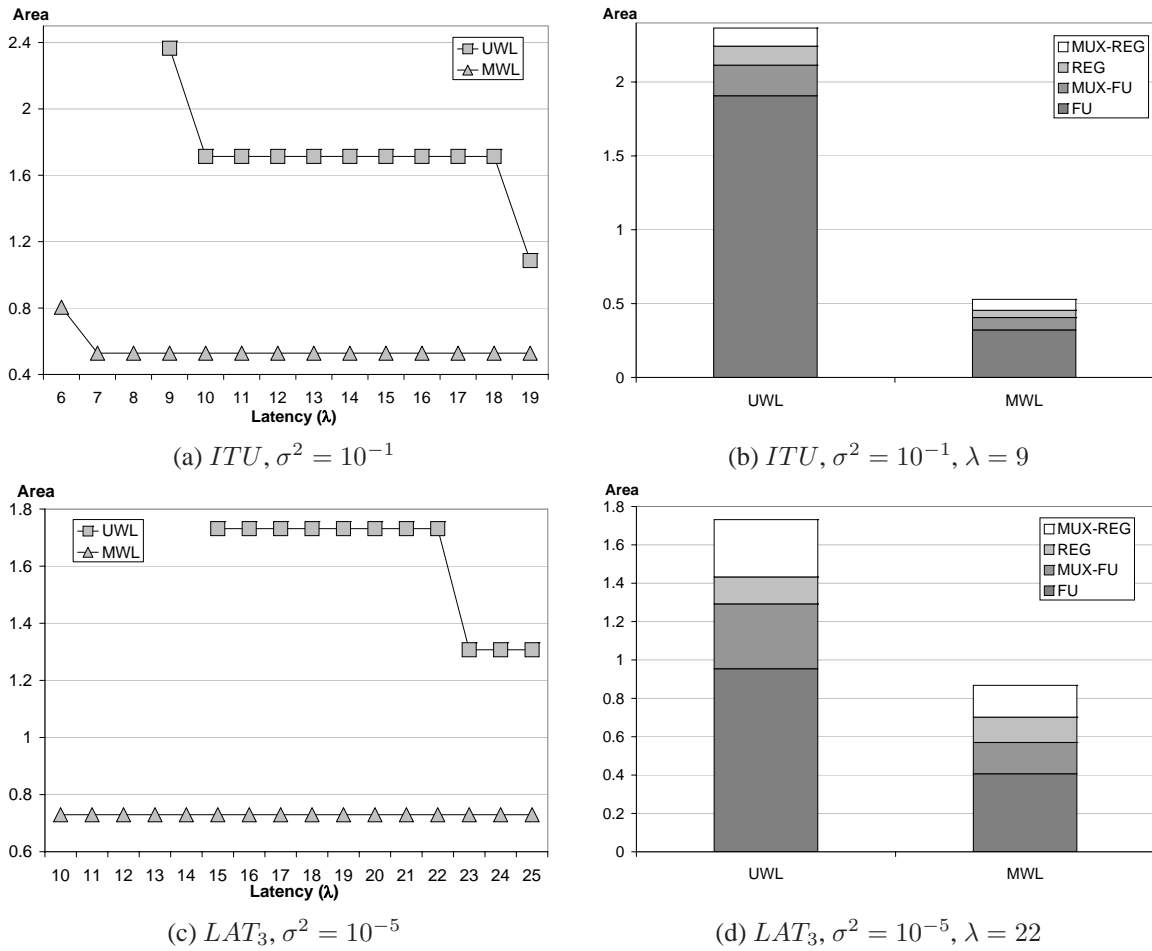


Figure 3.9: UWL vs. MWL: homogeneous implementations (I)

3.4.1. Uniform word-length vs. multiple word-length synthesis: Homogeneous architectures

Fig. 3.9 and Fig. 3.10 display results on the comparison of UWL vs. MWL synthesis using a homogeneous-resource architecture (i.e. only LUT-based resources). Note that the subfigures are arranged in couples, which are related to the same benchmark. The left subfigures depict the area vs. latency curves for a particular output noise constraint (Fig. 3.9-a,-c and Fig. 3.10-a,-c), while the right subfigures contain the detailed resource distribution graph of a particular point of its counterpart (Fig. 3.9-b,-d, and Fig. 3.10-b,-d). Let us define

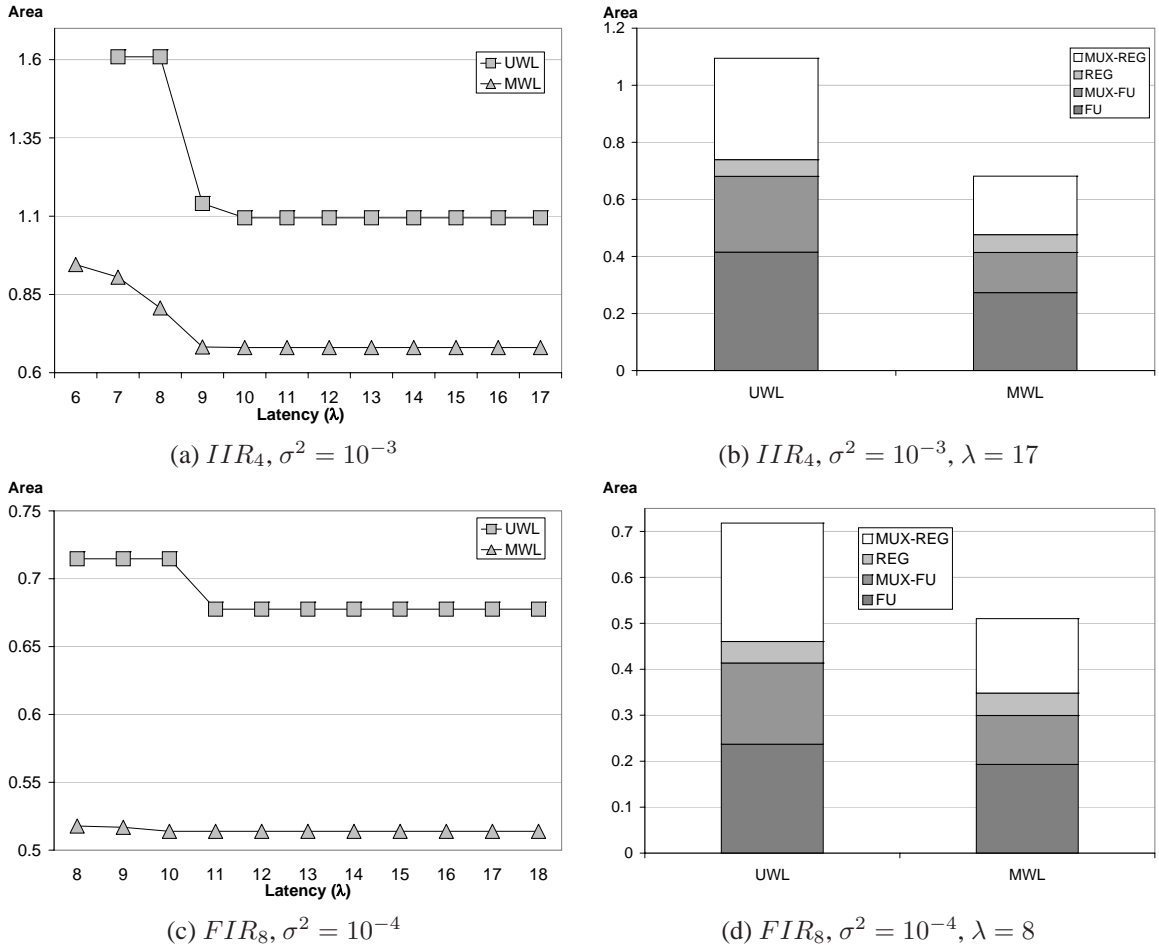


Figure 3.10: UWL vs. MWL: homogeneous implementations (II)

$\lambda_{min}^{UWL-HOM}$ as the minimum latency attainable for an UWL-homogenous implementation of an algorithm, and $\lambda_{min}^{MWL-HOM}$ the equivalent for an MWL-homogenous implementation. The latency used for the experiments ranges from $\lambda_{min}^{MWL-HOM}$ to $\lambda_{min}^{UWL-HOM} + 10$.

Fig. 3.9-a and Fig. 3.9-b contain the implementation results of the *ITU* benchmark with an output noise variance of 10^{-1} . Fig. 3.9-a depicts how both the UWL and MWL areas decrease as long as the latency increases. This is expected since the greater the latency the greater the chance of FU reuse. The comparison of the two implementation curves yields that the improvement obtained by means of using an MWL approach ranges from 51% to 77%. Also, the minimum latency that each implementation achieves differs considerably. The fine-

grain trade-off between area and quantization noise performed by the MWL approach allows important area reductions when compared to the UWL approach. Fig. 3.9-b displays the detailed resource distribution for the *ITU* UWL and MWL implementations correspondent to $\sigma^2 = 10^{-1}$ and $\lambda = 9$. The overall area savings are 77%, and it is due to the fact that the word-lengths of the majority of signals, which impact on FUs, multiplexers and registers, have been highly reduced: FUs' area has been reduced 83%, FU's multiplexers 59%, registers 62% and registers' multiplexers 39%. It is important to highlight that the area due to multiplexers and registers, although smaller than the FUs' area, makes up a significant part of the total area (20% for UWL and 39% for MWL). Hence the importance of including its cost within the optimization loop. This is analyzed in subsection 3.4.4.

The other benchmarks also show large area improvements: LAT_3 up to 49% (Fig. 3.9-c), IIR_4 up to 49% (Fig. 3.10-a) and FIR_8 up to 28% (Fig. 3.10-c). As observed in the detailed resource distribution subfigures (Fig. 3.9-d, and Fig. 3.10-b,-d), the area of the majority of the resources has been highly decreased. Also, it is noted that the percentage of area devoted to multiplexation and data storing is high in proportion to the overall implementation area. The minimum latency is also improved (see Fig. 3.9-a,-b and 3.10-a).

Table 3.3 contains the implementation results for all the benchmarks corresponding to three different quantization noise scenarios. For each quantization scenario the latency ranges from $\lambda_{min}^{UWL-HOM}$ to $\lambda_{min}^{UWL-HOM} + 10$, and the minimum, maximum and mean values of the area improvements obtained by the MWL implementations in comparison to the UWL implementations are computed. The first column in the table contains the name of the benchmark. The second, the output noise variance. And the third column contains area improvement values. The last row holds the minimum, maximum and average improvements considering all results simultaneously.

The area improvements obtained are remarkable: *ITU* obtains a mean improvement up to 77.66 %; LAT_3 up to 49.89 %; IIR_4 up to 64.99 %; FIR_8 up to 47.68 %. Note that the

Table 3.3: UWL vs. MWL for homogeneous architectures.

Bench.	σ_q^2	Area improvement (%)		
		min	max	mean
<i>ITU</i>	10^{-1}	51.36	77.66	68.32
	10^{-2}	48.44	76.31	66.41
	10^{-3}	46.51	75.40	65.13
<i>LAT₃</i>	10^{-3}	44.07	44.07	44.07
	10^{-4}	33.66	33.66	33.66
	10^{-5}	33.62	49.89	45.42
<i>IIR₄</i>	10^{-3}	37.85	49.86	39.69
	10^{-4}	34.30	63.55	50.65
	10^{-5}	37.08	64.99	52.72
<i>FIR₈</i>	10^{-3}	40.28	47.68	43.54
	10^{-4}	24.16	28.10	25.14
	10^{-5}	22.04	25.73	22.82
<i>All</i>		22.04	77.66	46.46

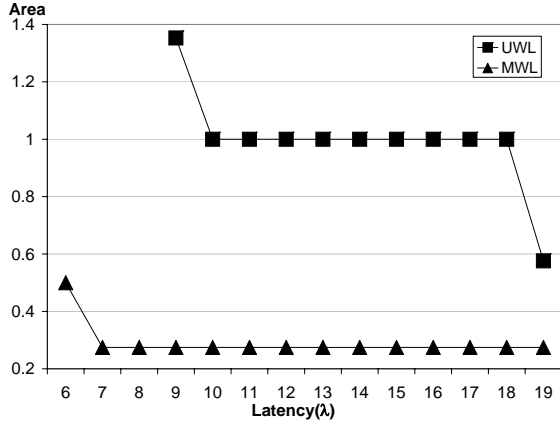
minimum improvements obtained for all benchmarks are quite close to both the maximum and the mean. The average improvement obtained is 46.46% and the maximum achieved is 77.66%. The results clearly show that an MWL-based HLS approach achieves significant area reductions.

Regarding latency, the minimum latency achievable by UWL implementations is reduced in average a 22% by means of MWL HLS.

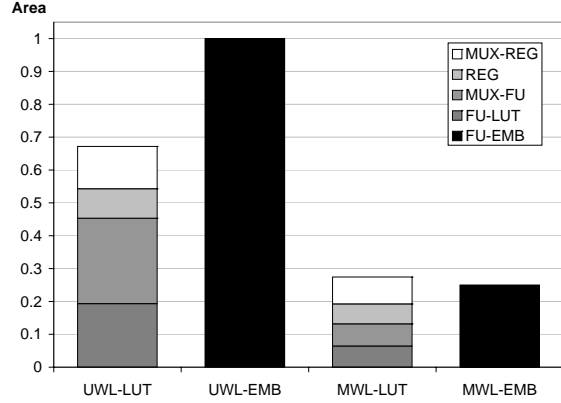
3.4.2. Uniform word-length vs. multiple word-length synthesis: Heterogeneous architectures

Fig. 3.11 and Fig. 3.12 contain results on the comparison of UWL vs. MWL synthesis using an heterogeneous-resource architecture (i.e. both LUT-based and embedded resources present). The arrangement of figures is similar to that of the previous subsection. Now, the latency ranges from $\lambda_{min}^{MWL-HET}$ to $\lambda_{min}^{UWL-HET} + 10$ (*HET* indicates heterogeneous

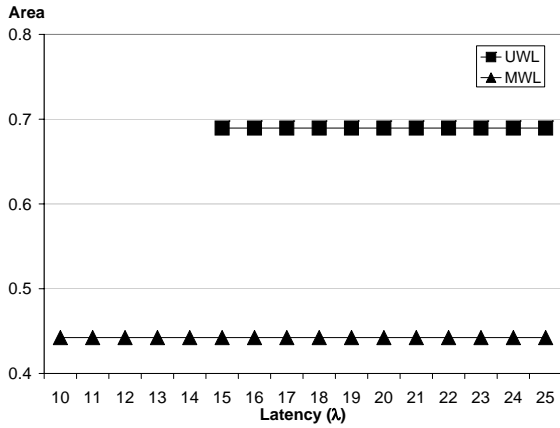
implementations).



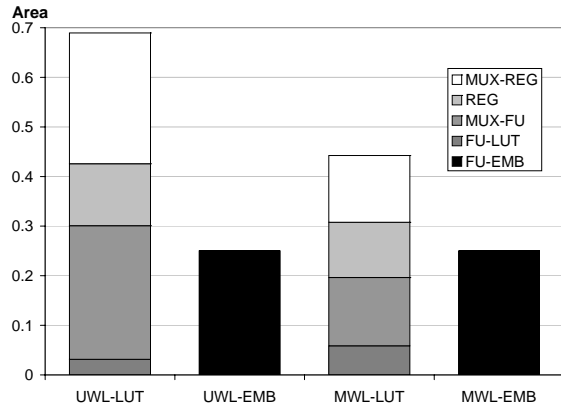
(a) $ITU, \sigma^2 = 10^{-2}$



(b) $ITU, \sigma^2 = 10^{-2}, \lambda = 10$



(c) $LAT_3, \sigma^2 = 10^{-4}$



(d) $LAT_3, \sigma^2 = 10^{-4}, \lambda = 15$

Figure 3.11: UWL vs. MWL: heterogeneous implementations (I)

Fig. 3.11-a,-c and Fig. 3.12-a,-c contain the implementation area vs. latency curves. The graphs clearly show how the area is reduced by means of an MWL synthesis: ITU up to 79% (Fig. 3.11-a), LAT_3 up to 35% (Fig. 3.11-c), IIR_4 up to 40% (Fig. 3.12-a), and FIR_8 up to 26% (Fig. 3.12-b). The detailed resource distribution in Fig. 3.11-b,-d and Fig. 3.12-b,-d shows how the majority of resources are decreased, and in particular the embedded multipliers and the FUs' multiplexers are clearly optimized. For instance, the ITU resource distribution for $\sigma^2 = 10^{-2}$ and $\lambda = 10$ (Fig. 3.11-b) shows an overall area reduction of 72%. The LUT-based resources are reduced 59% (LUT-based FUs' area has been reduced 32%,

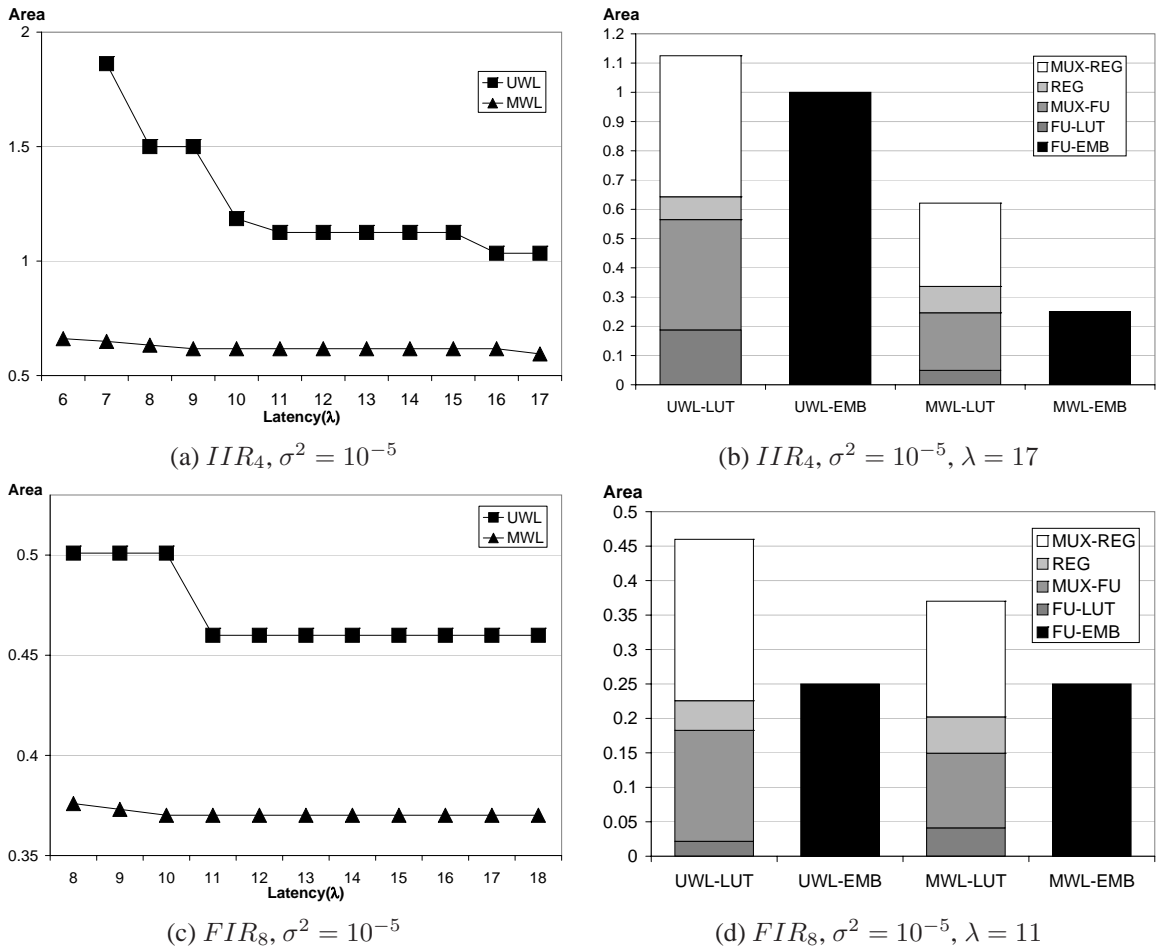


Figure 3.12: UWL vs. MWL: heterogeneous implementations (II)

FU’s multiplexers 74%, registers 32%, and registers’ multiplexers 36%); while the embedded FUs are reduced 75%.

Again, the figures show how the minimum latency can be highly improved by means of an MWL approach. Also, it can be seen that the LUT-based resources are devoted almost entirely to data multiplexing and storing.

Table 3.4 contains the implementation results of all the benchmarks corresponding to three different quantization noise scenarios. For each quantization scenario the latency ranges from $\lambda_{min}^{UWL-HET}$ to $\lambda_{min}^{UWL-HET} + 10$, and the minimum, maximum and mean area improvements obtained by the MWL implementations in comparison to the UWL imple-

Table 3.4: UWL vs. MWL for heterogeneous architectures.

Bench.	σ_q^2	$\ \hat{A}\ _\infty$ %			A_{LUT} %			A_{EMB} %		
		min	max	mean	min	max	mean	min	max	mean
<i>ITU</i>	10^{-1}	54.85	80.77	73.69	55.56	81.07	63.19	50.00	75.00	72.73
	10^{-2}	52.37	79.71	71.37	52.37	79.71	60.41	50.00	75.00	72.73
	10^{-3}	47.80	77.76	66.71	47.80	77.76	56.33	50.00	75.00	72.73
<i>LAT₃</i>	10^{-3}	48.87	48.87	48.87	48.87	48.87	48.87	0.00	0.00	0.00
	10^{-4}	35.84	35.84	35.84	35.84	35.84	35.84	0.00	0.00	0.00
	10^{-5}	31.70	31.70	31.70	31.70	31.70	31.70	0.00	0.00	0.00
<i>IIR₄</i>	10^{-3}	37.47	41.27	38.33	37.47	45.10	39.05	0.00	0.00	0.00
	10^{-4}	32.97	40.70	34.74	32.97	40.70	34.74	0.00	0.00	0.00
	10^{-5}	40.32	65.13	49.57	40.32	65.13	48.55	50.00	83.33	71.67
<i>FIR₈</i>	10^{-3}	32.45	38.01	33.97	38.79	43.83	39.68	0.00	0.00	0.00
	10^{-4}	27.53	32.83	28.62	27.53	32.83	28.62	0.00	0.00	0.00
	10^{-5}	19.53	26.12	21.17	19.53	26.12	21.17	0.00	0.00	0.00
<i>All</i>		19.53	80.77	44.88	19.53	81.07	42.76	0.00	83.33	24.03

mentations are computed considering ∞ -norm area, the LUT-based area and the embedded FUs area. The first column in the table contains the name of the benchmark. The second, the output noise variance applied. The third column contains the minimum, maximum and mean ∞ -norm area improvement values. The fourth column contains the minimum, maximum and mean values of the LUT-based resource area. And the last column contains the minimum, maximum and mean values of the embedded FUs area.

The area improvements obtained are considerable: *ITU* obtains up to 80.77%, *LAT₃* up to 48.87%, *IIR₄* up to 65.13%, *FIR₈* up to 38.01%. Note that the minimum improvements obtained for most of the benchmarks are again quite close to both the maximum and the mean.

The LUT-based area reductions are up to 81.07% for *ITU*, up to 48.87% for *LAT₃*, up to 65.13% for *IIR₄*, and up to 43.83% for *FIR₈*. The embedded resources are only reduced for benchmarks *ITU* (up to a 75.00%) and *IIR₄* (up to 83.33%). Benchmarks *LAT₃* and

FIR_8 use the minimum possible number of embedded resources (1 embedded multiplier), hence the 0% improvement.

Area improvements up to 80.77% are achieved. The average improvement is 44.88% for the overall area, 42.76% for the LUT-based resources, and 24.03% for the embedded resources. The results clearly show that an MWL-based approach for HLS leads to significant area reductions.

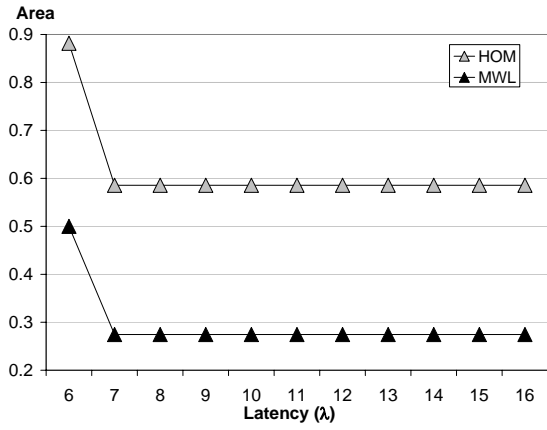
The latency analysis throws that the minimum UWL latency is reduced an average 19% by means of MWL HLS.

3.4.3. MWL synthesis: Heterogeneous vs. Homogeneous

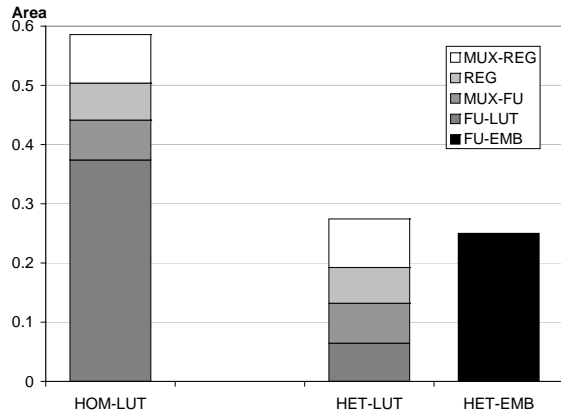
Fig. 3.13 and Fig. 3.14 display the results of the comparison of MWL implementations using homogeneous architectures vs. heterogeneous architectures. The arrangement of figures is similar to that of the previous subsections. The latency ranges from the $\lambda_{min}^{MWL-HOM}$ to $\lambda_{min}^{MWL-HET} + 10$.

Fig. 3.13-a and Fig. 3.13-b contain the implementation results for the *ITU* benchmark with an output noise variance of 10^{-2} . The comparison of the two implementation curves yields that the improvement obtained by using embedded resources ranges from 43% to 53%, since large multipliers can be now implemented using embedded multipliers. Fig. 3.13-b displays the detailed resource distribution for the *ITU* implementations correspondent to $\sigma^2 = 10^{-2}$ and $\lambda = 10$. The overall area savings are 53%, and it is mainly due to the area reduction of LUT-based FUs (99%).

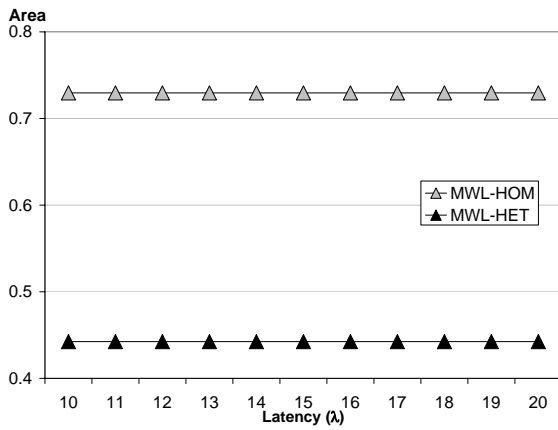
The area vs. latency graphs clearly show how area is reduced by means of using an heterogeneous architecture: LAT_3 up to 39% (Fig. 3.13-c), IIR_4 up to 48% (Fig. 3.14-b), FIR_8 up to 44% (Fig. 3.14-d). The detailed resource distribution in Fig. 3.13-b,-d and Fig. 3.14-b,-d shows that the main reason for the area improvement is the substantial area reduction of LUT-based FUs, due to the migration of LUT-based multipliers to embedded



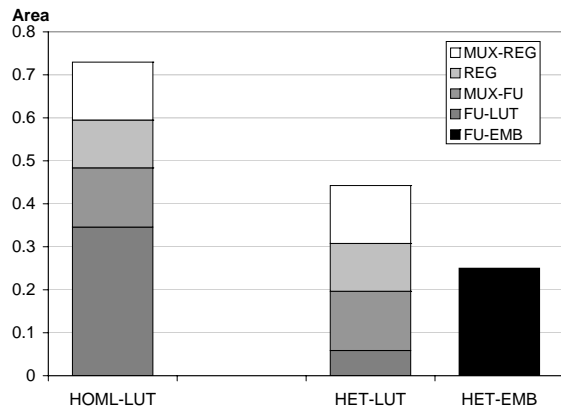
(a) $ITU, \sigma^2 = 10^{-2}$



(b) $ITU, \sigma^2 = 10^{-2}, \lambda = 106$



(c) $LAT_3, \sigma^2 = 10^{-4}$



(d) $LAT_3, \sigma^2 = 10^{-4}, \lambda = 15$

Figure 3.13: MWL synthesis: homogeneous vs. heterogeneous (I)

multipliers.

Table 3.5 contains the implementation results of all the benchmarks corresponding to three different quantization noise scenarios. For each quantization scenario the latency ranges from $\lambda_{min}^{MWL-HOM}$ to $\lambda_{min}^{MWL-HOM} + 10$, and the minimum, maximum and mean values of the area improvements, in terms of ∞ -norm, obtained by the MWL implementations in comparison to the UWL implementations are computed. The first column of the table contains the name of the benchmark. The second, the output noise variance applied. And, the third column contains the minimum, maximum and mean area improvement values.

The area improvements obtained are remarkable: ITU obtains up to 54.76%; LAT_3 up

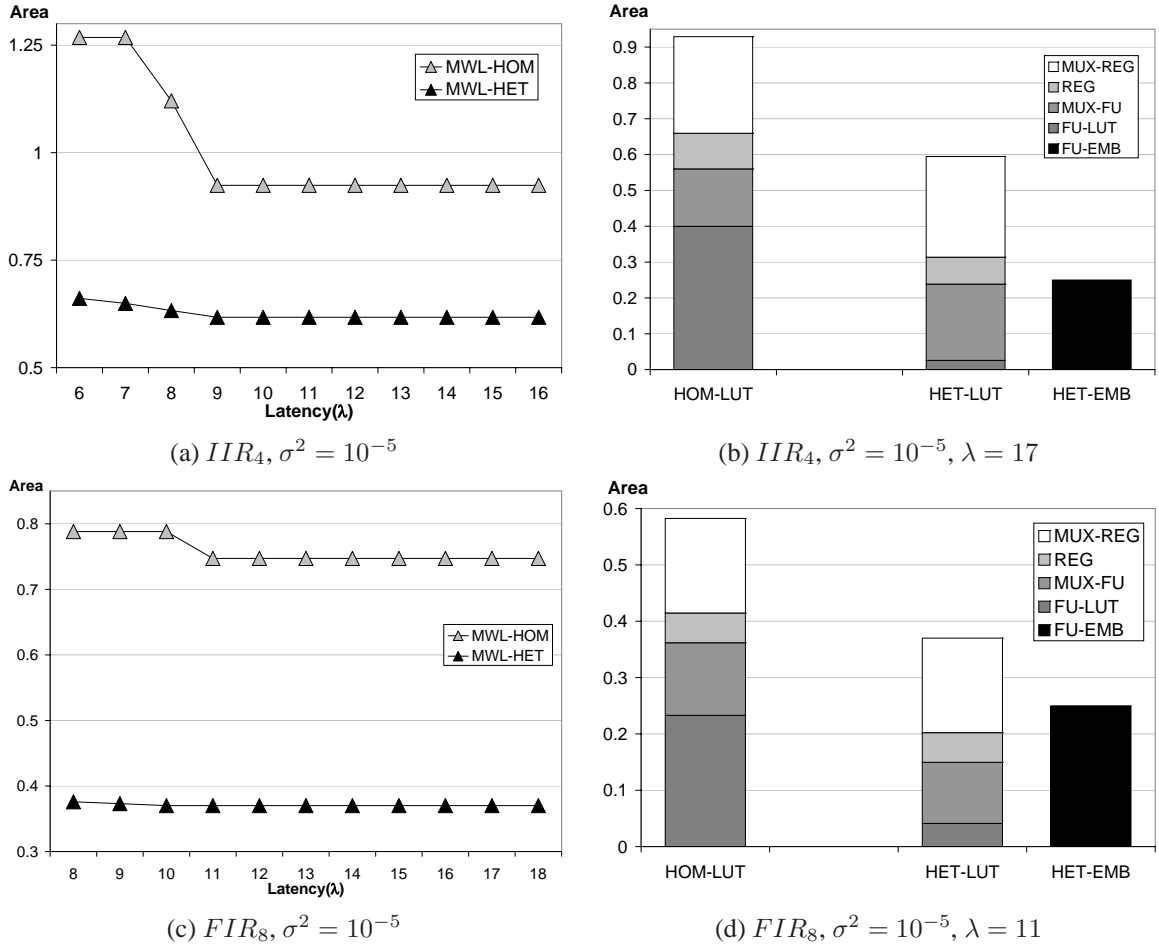


Figure 3.14: MWL synthesis: homogeneous vs. heterogeneous (II)

to 43.09%; IIR_4 up to 48.79%; FIR_8 up to 44.68%. Note that, again, the minimum improvements obtained for all benchmarks are quite close to both the maximum and the mean. Area improvements up to 54.76% are achieved, being the average improvement 40.23%. The results clearly show that the inclusion of embedded resources within HLS leads to highly optimized DSP implementations.

Regarding latencies, the minimum latency achievable by both homogeneous and heterogeneous implementations is the same for the experiments performed. This is due to the fact that the latency of resources are very similar in the particular conditions used for the tests. The same experiments presented in this section were repeated increasing the constant word-

Table 3.5: MWL synthesis: homogeneous vs. heterogeneous architectures.

Bench.	σ_q^2	$\ \hat{A}\ _\infty$		% mean
		min	max	
<i>ITU</i>	10^{-1}	40.73	52.69	51.60
	10^{-2}	43.29	53.98	53.01
	10^{-3}	50.45	54.76	54.32
<i>LAT₃</i>	10^{-3}	42.68	43.09	42.72
	10^{-4}	39.36	39.36	39.36
	10^{-5}	38.73	39.74	38.82
<i>IIR₄</i>	10^{-3}	34.83	44.77	36.04
	10^{-4}	32.92	48.23	35.96
	10^{-5}	33.21	48.79	35.79
<i>FIR₈</i>	10^{-3}	21.42	41.46	28.37
	10^{-4}	27.02	44.68	33.24
	10^{-5}	27.46	44.62	33.52
<i>All</i>		21.42	54.76	40.23

length to 16 bits, obtaining that heterogeneous implementations reduced 7% the minimum latency in contrast to homogeneous implementations.

3.4.4. Effect of registers and multiplexers

In this subsection the effect of including the cost of registers and multiplexers within the optimization loop is investigated. As in the previous experiments, the analysis is performed implementing the benchmarks using different noise and latency constraints. Before HLS is applied a gradient-descent quantization [CSL02] is applied according to the given noise constraint. The comparison is done by using algorithm 3.1 to perform the HLS using two different area cost estimation solutions: (i) algorithm 3.2, which it is referred to as the *complete* area estimation algorithm, and (ii) a simplified version of algorithm 3.2 (*simplified* area estimation algorithm) where the cost of registers and multiplexers is neglected. When the simplified area estimation is used, the cost of registers and multiplexers is included after the

Table 3.6: Complete vs. simplified cost estimation: Area improvement (%).

Arch.	Bench.	Area improvement		
		min	max	mean
HOM	<i>ITU</i>	0.00	0.95	0.30
	<i>LAT₃</i>	0.71	3.50	1.53
	<i>IIR₄</i>	0.00	5.35	1.26
	<i>FIR₈</i>	0.00	1.77	0.31
HET	<i>ITU</i>	0.00	25.85	1.89
	<i>LAT₃</i>	1.15	5.77	2.52
	<i>IIR₄</i>	0.00	35.57	8.09
	<i>FIR₈</i>	0.00	8.11	0.83
<i>All</i>		0.00	35.57	2.09

Table 3.7: Complete vs. simplified cost estimation: Optimization time.

Arch.	Bench.	σ^2	Complete (secs)	only-FUs (secs)
HOM	<i>ITU</i>	10^{-1}	24.52	15.27
	<i>LAT₃</i>	10^{-3}	139.90	74.83
	<i>IIR₄</i>	10^{-3}	118.10	39.64
	<i>FIR₈</i>	10^{-3}	115.55	56.05
HET	<i>ITU</i>	10^{-1}	43.17	13.91
	<i>LAT₃</i>	10^{-3}	159.30	112.80
	<i>IIR₄</i>	10^{-3}	115.6	57.00
	<i>FIR₈</i>	10^{-3}	138.5	100.09

optimization loop has finished its execution, using the complete area estimation (Alg. 3.2).

Table 3.6 contains the results of this analysis. The latencies range from λ_{min}^{ARCH} to $\lambda_{min}^{ARCH} + 10$, where *ARCH* refers to the type of FPGA architecture used (homogeneous or heterogeneous). The noise constraints are the same used in the previous subsection (three σ^2 for each benchmark), though the results have been combined into a single row. The first column contains the type of FPGA architecture. The second column indicates the benchmark used. And the fourth column contains the minimum, maximum and average area improve-

ment obtained by the complete area estimation synthesis in contrast to the simplified area estimation synthesis. The last row includes the minimum, maximum and mean improvements for all benchmarks.

The average improvements for the different benchmarks range from 0.00% to 8.09%, being the overall average improvement of 2.09%. The maximum improvement found is 35.57%. These results clearly show that failing to include the cost of registers and multiplexer during the optimization procedure can lead to unwanted area penalties.

Table 3.7 contains information regarding typical optimization times. The computer used to carry out the experiments has a 1.66 GHz Intel Core Duo processor and 1 GB of RAM. The latency constraint for these experiments was set to $\lambda = \lambda_{min} + 3$ clock cycles, while the noise constraint is indicated in the third column of the table. The last two columns show the optimization times required for the complete and simplified approaches.

The simplified approach requires optimization times much smaller than those of the complete approach. This is due to the fact that the optimization problem has been simplified, thus it is necessary to perform less iterations to converge to a solution. Also the execution time of each iteration is smaller since the tasks of register binding and multiplexer selection are not required. However, the maximum error produced by the simplified approach (from 1.77% to 35.57%) indicates that the extra time required by the complete approach might be worth it. It is interesting to see how the optimization times seems to be related to the number of nodes of each algorithm (see table 3.2).

3.5. Conclusions

The HLS of MWL DSP algorithms has been presented in this chapter. A novel HLS procedure is presented able to consider many aspects that have been neglected in previous approaches found in the literature. These aspects are as follows:

- Complete set of datapath resources that includes FUs, registers and multiplexers.
- MWL-oriented set of resources that include word-length dependent resource cost models: area and latency.
- FPGA-oriented set of resources composed of both LUT-based and embedded resources.
- Optimizer able to handle such a complex set of resources. This implies the use of word-length specific SA movements as well as a novel area metrics for heterogeneous-architecture FPGA resources.

The results show that:

- An MWL HLS approach obtains average improvements up to a 77% with an average value of 46% for homogeneous architectures. Minimum latencies are improved 22%.
- An MWL HLS approach obtains improvements of up to a 80% with an average value of 44% for heterogeneous architectures. Minimum latencies are improved 19%.
- The use of a heterogeneous-architecture oriented MWL HLS vs. a homogeneous-architecture oriented MWL HLS provides improvements of up to 54% with an average value of 40%.
- The use of a complete area cost estimation that simultaneously considers FUs, registers and multiplexer within an MWL-based optimization procedure leads to improvements of up to 35%.

CHAPTER 4

WORD-LENGTH ALLOCATION

This chapter addresses the problem of word-length allocation. WLA is the procedure that selects the word-lengths of the different signals of a circuit in order to optimize a particular design cost, while complying with a given output noise constraint. In this work, as in many others, WLA is divided into two main stages: *scaling* (MSB selection) and word-length selection (LSB selection). The scaling stage fixes the position of the MSB with respect to the binary point. The word-length selection stage finds the proper set of signal word-lengths, that is the number of LSBs, which minimizes cost. During word-length selection it is necessary to estimate the output noise. The computational complexity of this stage is mainly determined by the optimization procedure used (i.e. optimal or heuristic), but it can also be highly affected by the error estimation technique used (i.e. analytical or simulation-based). Thus, using fast error estimation techniques enables the application of complex optimization procedures that lead to highly reduced implementation costs. A fast and accurate novel affine arithmetic-based error estimation technique applicable to both LTI and non-linear systems is presented in this chapter.

Traditionally, WLA is a task prior to HLS. In this chapter the effect of using different WLA optimization techniques on the HLS implementations are analyzed.

The chapter is divided as follows. Section 4.1 introduces some key WLA concepts. Next, in section 4.2, scaling and error estimation are dealt with. An affine arithmetic-based technique to estimate the output error for both LTI and non-linear systems is presented. Section 4.3 deals with word-length selection techniques. Results on the accuracy of the error estimation method proposed, as well as on the effect of WLA on the HLS implementations are presented in section 4.4. Finally, the conclusions are drawn in section 4.5.

4.1. Word-length Allocation

The starting point of WLA is a graph $G_{SFG}(V, S)$, already introduced in the previous chapter, that contains information about the signal fixed-point formats and the data dependencies, which are both necessary to estimate the quantization error at the output of the algorithm. Initially, the fixed-point format of signals (i.e. the (p, n) couples) are unknown and it is the task of WLA to find a suitable set of these that minimizes cost. The fixed-point format determines the quantization error generated by a quantized signal, and the data dependency, along with the input data, the effect of that error on the output's accuracy. Also, the fixed-point formats determine the number of bits of each signal, and therefore the size of the resources (e.g. FUs, registers and multiplexers) that process them. The size of the resource ultimately determines its area, delay and power costs. During WLA, the optimization is guided by means of the cost and the output error obtained from the different fixed-point formats tried through successive iterations.

Fig. 4.1 depicts the WLA approach adopted in this work. WLA is composed of the stages of scaling, which determines the set of p , and word-length selection, which determines the set of n . This subdivision allows to simplify WLA while still providing significant cost

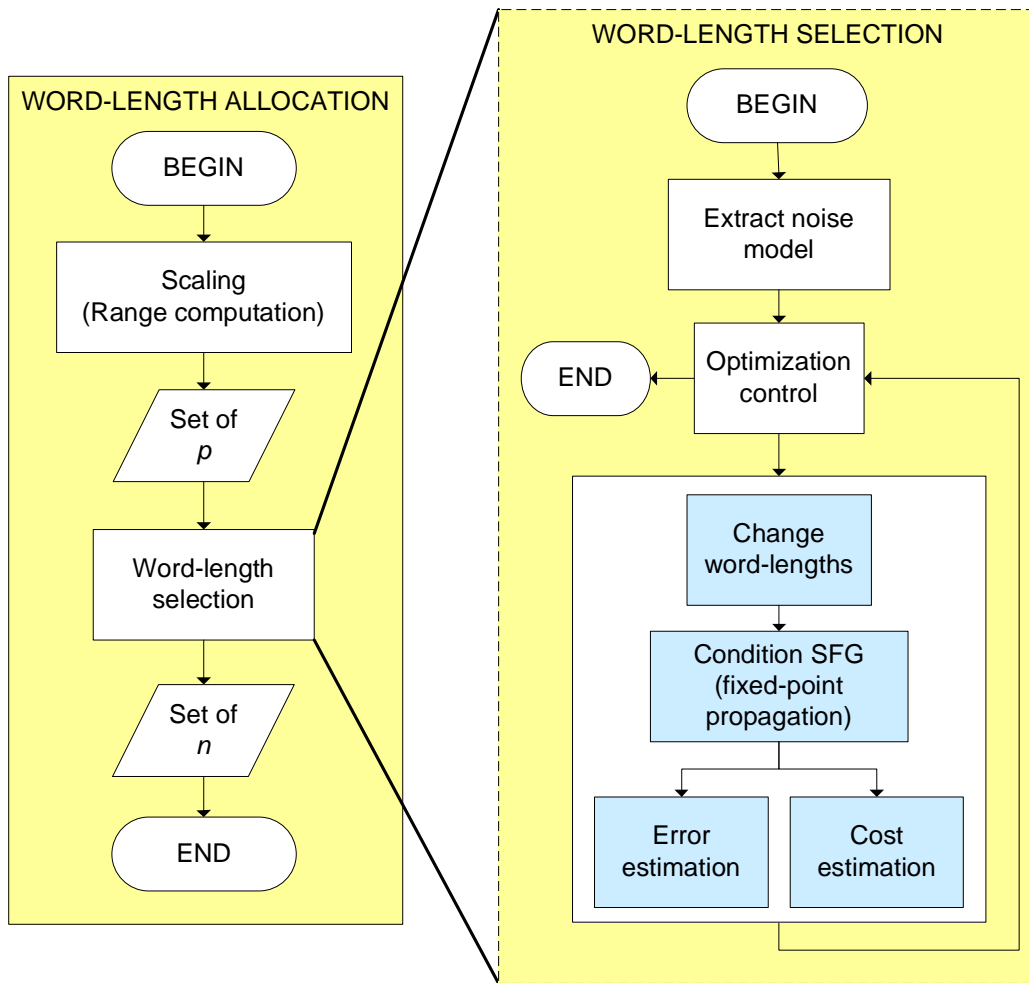


Figure 4.1: Word-length allocation diagram.

reductions.

A wrap-around scaling strategy is adopted since it requires less hardware than other approaches (i.e. saturation techniques). After scaling, the values of p are the minimum possible values that avoid the overflow of signals or, at least, those that reduce the likelihood of overflow to a negligible value). Among the different options (simulation-based, analytical, etc.) both an AA-based approach, for LTI algorithms, and a simulation-based approach, for non-linear algorithms, are considered in this work (see subsection 4.2.2).

Once scaling is performed, the values of p can be fixed during word-length selection. The

right side of Fig. 4.1 shows a diagram of the basic blocks forming word-length selection. In particular, the diagram refers to a heuristic approach. Basically, word-length selection iterates trying different word-length (i.e. n) combinations until the cost is minimized. Any time the word-length of a signal, or a group of signals, is changed, the word-lengths must be propagated throughout the graph, task referred to as graph *conditioning* [CCL03a]. The *optimizer control* block selects the size of the word-lengths (set of n) using the values of the previous error and cost estimations and decides when the optimization procedure has finished. The first operation in the diagram is the extraction of the quantization noise model. The role of this operation is to generate a model of the quantization noise at the output due to the fixed-point format of each signal. This enables to perform a quick error estimation within the optimization loop, therefore, leading to a wider design space exploration. This method is not the most popular due to the difficulty in finding a proper noise model, especially for non-linear systems, and a simulation-based approach is usually taken instead. For instance, the quantization of an 8-bit coefficients IIR biquad filter with a SQNR (signal to quantization noise ratio) of 30 dB requires approximately 300 noise estimates. Thus, a simulation-based WLA would require 300 bit-true simulations. As long as the complexity of the algorithms increases, WLA time reaches prohibitive values. Hence, the importance of using fast error estimation techniques. In subsection 4.2.3 an error model able to handle LTI as well as some non-linear systems is presented.

For simplification's sake, some fixed-point information inherent to quantized signals from G_{SFG} was omitted in the previous chapter. First, let us assume that signals have an index associated to them so $S = \{s_0, \dots, s_{|S|-1}\}$. Any signal $s_i \in S$ contains information about its mathematical precision by means of the couples (p^{pre}, n^{pre}) and (p, n) : the first couple indicates the scaling and word-length of the signal before quantization; the latter indicates the scaling and word-length of the signal after quantization. Fig. 4.2 depicts the situation where two signals a and b with format $(p, n) = (2, 8)$ are multiplied and the result

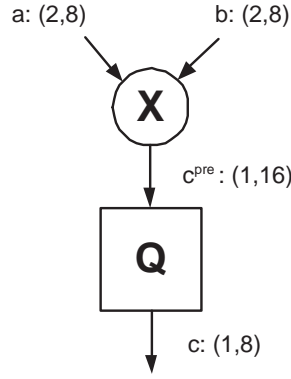


Figure 4.2: Signals' fixed-point information

of the multiplication is quantized. The pre-quantization fixed-point parameters of the output c are $(p^{pre}, n^{pre}) = (1, 16)$ and the parameters after the quantization are $(p, n) = (1, 8)$. The word-length of signal c has been reduced from 16 bits (n^{pre}) to 8 bits (n). These two couples are used to compute the parameters of the noise generated by each signal. Since scaling is assumed to be fixed, the values of p^{pre} and p are the same for a particular signal. Here, the noise model proposed in [CCL99] (which is referred to as the *discrete noise model*) is adopted, which is an extension of the traditional modeling of quantization error as an additive white noise [OW72] (referred to as the *continuous noise model*). Given a signal s_i with scaling p_i and n_i^{pre} bits quantized to n_i bits the variance and mean of the quantization noise produced are:

$$\sigma_i^2 = \frac{2^{2p_i}}{12} \left(2^{-2n_i} - 2^{-2n_i^{pre}} \right) \quad (4.1)$$

$$\mu_i = 2^{p_i-1} \left(2^{-n_i} - 2^{-n_i^{pre}} \right). \quad (4.2)$$

Both models assume that quantization noise can be replaced by an additive uniform noise, however, the latter assumes that the signal has infinite precision before quantization. The error of the continuous model can be as high as 200% when n is close to n^{pre} .

Note that both the continuous and discrete quantization noise models assume that the quantization noise does not depend on the particular values of signal s_i but only on the fixed-point parameters (p_i, n_i) and (p_i^{pre}, n_i^{pre}) . This assumption can be considered valid if the dynamic range of the signal is much larger than the range of the quantization error. However, the noise of a signal s_i traverses the signal flow graph G_{SFG} from s_i to the output (or outputs), so the effect of the quantization of s_i at the output of the algorithm depends on the operations that form the path from s_i to the output and on the data that traverses this path (which also includes the constants used in gains). In LTI systems the variance, mean and power of the quantization noise at the output of the system in permanent state can be estimated in terms of σ_i and μ_i by means of:

$$\begin{aligned}\sigma_o^2 &= \sum_{i=0}^{|S|-1} \sigma_i^2 \cdot \frac{1}{2\pi} \int_{-\pi}^{\pi} |G_i(e^{j\Omega})|^2 d\Omega \\ &= \sum_{i=0}^{|S|-1} \sigma_i^2 \cdot \sum_{j=0}^{\infty} g_i^2[j]\end{aligned}\quad (4.3)$$

$$\begin{aligned}\mu_o &= \sum_{i=0}^{|S|-1} \mu_i \cdot G_i(1) \\ &= \sum_{i=0}^{|S|-1} \mu_i \cdot \sum_{j=0}^{\infty} g_i[j]\end{aligned}\quad (4.4)$$

$$P_o = \sigma_o^2 + (\mu_o)^2, \quad (4.5)$$

where G_i and g_i are the transfer function and the impulse response from signal s_i to output s_o . This expression can be formulated more compactly using vectors $\boldsymbol{\sigma}^2$, \boldsymbol{v} , $\boldsymbol{\mu}$ and \boldsymbol{m} .

$$\sigma_o^2 = \boldsymbol{\sigma}^2 \cdot \mathbf{v}^T \quad (4.6)$$

$$\mu_o = \boldsymbol{\mu} \cdot \mathbf{m}^T \quad (4.7)$$

$$P_o = \boldsymbol{\sigma}^2 \cdot \mathbf{v}^T + (\boldsymbol{\mu} \cdot \mathbf{m}^T)^2 \quad (4.8)$$

$$\boldsymbol{\sigma}^2 \equiv \langle \sigma_0^2, \dots, \sigma_{|S|-1}^2 \rangle \quad (4.9)$$

$$\boldsymbol{\mu} \equiv \langle \mu_0, \dots, \mu_{|S|-1} \rangle \quad (4.10)$$

$$\mathbf{v} \equiv \left\langle \sum_{j=0}^{\infty} g_0^2[j], \dots, \sum_{j=0}^{\infty} g_{|S|-1}^2[j] \right\rangle \quad (4.11)$$

$$\mathbf{m} \equiv \left\langle \sum_{j=0}^{\infty} g_0[j], \dots, \sum_{j=0}^{\infty} g_{|S|-1}[j] \right\rangle \quad (4.12)$$

Note that \mathbf{v} and \mathbf{m} can be computed by means of a graph analysis, and once they are determined, the power of the output noise can be estimated if $\boldsymbol{\sigma}^2$ and $\boldsymbol{\mu}$ are known. In Fig. 4.1, \mathbf{v} and \mathbf{m} are computed in the block that extracts the noise parameters, and eqn. 4.6, or eqn. 4.8, would be used in the error estimation block. A similar approach has been applied in [CCL03a] and [MS02a].

Similar expressions to eqns. 4.6-4.8 have been obtained for differentiable non-linear systems by means of applying the perturbation theory. These methods are based on analyzing the effect of small perturbations, that is the quantization of signals, in the outputs of a system. In [Con03] a simulation-based method which produces an estimation of the variance of the noise is presented. The proposed method is fast, since it requires $|S|$ simulations to extract the noise parameters, however the contributions of the signal quantization noises at the output are supposed to be independent, which is a strong assumption difficult to meet. Also, the accuracy of the method is not contrasted with any bit-true data. Moreover, while the noise estimation only regards the variance of the noise, SNR vs. implementation area results are presented, which would imply the estimation of the power of the quantization er-

ror, and not only the variance. An alternative method is presented in [SB04a], where $|S|^2/2$ simulations as well as a curve fitting technique (with $|S|^2/2$ variables) are required to parameterize quantization noise. The method is tested with an LMS adaptive filter and the accuracy is checked graphically. Finally, in [MRSS04, RMHS06] a similar approach is followed for non-recursive differentiable algorithms and it is adapted to recursive systems by means of manual intervention.

In the next subsection an AA-based method able to parameterize the quantization output noise of both LTI and differentiable non-linear systems (both non-recursive and recursive) by means of a single AA simulation, is presented.

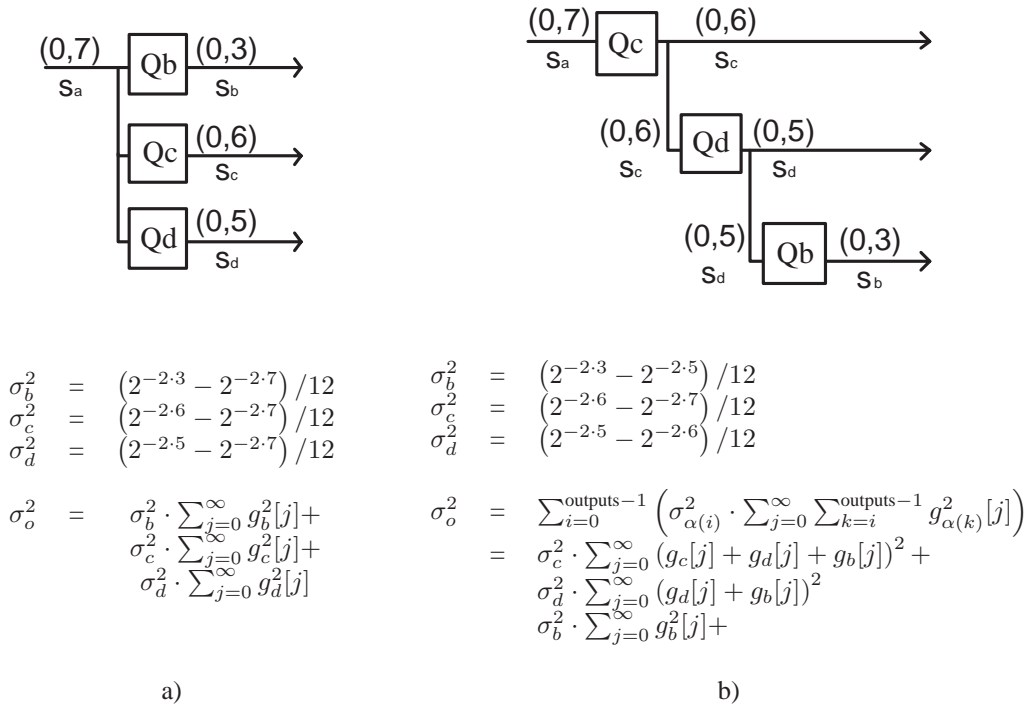


Figure 4.3: Quantization of forks: a) naive approach; b) accurate cascade model

Another important contribution that this work accounts for is the description of the behavior of quantization noise in forks [CCL03a]. Let us suppose that a signal is quantized, and then it is used as an input to several operations. It is obvious that the quantization noise is only one, no matter the number of operations that use that signal, so it would be an important

mistake to treat each input as an independent signal, since that leads to the addition of several noise sources instead of a single one. In a similar fashion, let us imagine that a signal is fed to different operations, but each operation's input requires different word-lengths. Again, it would be a mistake to ignore the correlation between noise sources, as in Fig. 4.3.a, where the quantization of a fork with input signal a and outputs b , c and d is handled treating each output as uncorrelated. The correct treatment to forks was explained in [CCL03a] and it requires the use of the cascade noise model (see Fig. 4.3.b). The application of this model implies the descending reordering of the outputs in terms of their word-lengths (i.e. n) and it considers that only one signal is quantized. Concretely, in Fig. 4.3-b, the ordering function $\alpha : \mathbb{N} \rightarrow \mathbb{N}$ has the following values:

$$\alpha(i) = \begin{cases} c & \text{if } i = 0 \\ d & \text{if } i = 1 \\ b & \text{if } i = 2 \end{cases} \quad (4.13)$$

The final expression for the noise variance at the output of the system s_o (σ_o^2) clearly differs from one approach to the other. The implications of the cascade model not only are those related to the noise estimation accuracy but they also impact on the way the noise parameterization is performed. For instance, the elements of vectors \mathbf{v} and \mathbf{m} (see eqn. 4.11 and eqn. 4.12) related to the outputs of forks depend on the particular values of their word-lengths, so they are not constant vectors. These vectors should be modified after the conditioning task (see Fig. 4.1), which seems to complicate the estimation method. In the next subsection a fast method to *update* these vectors is presented. It will be also presented a novel way to apply the cascade model to non-linear systems. This fact has been neglected in the existent literature handling estimator for non-linear systems [Con03, SB04a, RMHS06].

4.2. Scaling and error estimation

4.2.1. Affine arithmetic

Affine Arithmetic (AA) [SF97] is an extension of Interval Arithmetic (IA) [Hay03] aimed to perform fast and accurate computation of the ranges of the signals of a mathematical description of an algorithm. Its main feature is that it automatically cancels the linear dependencies of the included uncertainties along the computation path, thus avoiding the over-sizing produced by IA approaches [Lóp04]. It has been applied to both scaling computation [Lóp04, LCNT07, LGC⁺06] and word-length allocation [Lóp04, LGC⁺06, FCR03].

The mathematical expression of a given affine form is

$$\hat{x} = x_0 + \sum_{i=1}^N x_i \epsilon_i \quad (4.14)$$

where x_0 is the central value of \hat{x} , and ϵ_i and x_i are its i -th noise term identifier and amplitude, respectively. In fact, $x_i \epsilon_i$ represents the interval $[-x_i, +x_i]$. Affine operations are those which operate affine forms and produce an affine form as a result. Given the affine forms \hat{x} , \hat{y} and $\hat{c} = c_0$ the affine operations are

$$\hat{x} \pm \hat{c} = x_0 \pm c_0 + \sum_{i=1}^N x_i \epsilon_i \quad (4.15)$$

$$\hat{x} \pm \hat{y} = x_0 \pm y_0 + \sum_{i=1}^N (x_i \pm y_i) \epsilon_i \quad (4.16)$$

$$\hat{c} \cdot \hat{x} = c_0 x_0 + \sum_{i=1}^N c_0 x_i \epsilon_i \quad (4.17)$$

These operations suffice to model any LTI algorithm. However, it is also interesting to include the multiplication since it is a common operation in DSP algorithms. Although it is

not an affine operation, it can be approximated as one as shown in 4.18.

$$\hat{x} \cdot \hat{y} \approx x_0 y_0 + \sum_{i=1}^N (x_0 y_i + y_0 x_i) \epsilon_i + \sum_{i=1}^N \sum_{j=1}^N (x_i y_j) \epsilon_i^j \quad (4.18)$$

In some cases, it could be desirable to simplify this expression even further.

$$\hat{x} \cdot \hat{y} \approx x_0 y_0 + \sum_{i=1}^N (x_0 y_i + y_0 x_i) \epsilon_i \quad (4.19)$$

Given an affine form \hat{x} the affine form that contains all its possible values, that is its range, is defined as

$$range(\hat{x}) = x_0 + \left(\sum_{i=1}^N |x_i| \right) \epsilon \quad (4.20)$$

The application of AA to range computation and quantization noise estimation are developed in the next two subsections.

4.2.2. Scaling

In [Lóp04] it was proved that in feedback systems the scaling computation can be calculated by means of performing a simulation in time of the algorithm, where at each time step each input is fed with a new affine form which corresponds to its range. The range of each algorithm's signal can be obtained by computing the range of the affine form that the signal holds at each time step, taking the maximum range obtained during the simulation as the definitive range.

In LTI signals an AA simulation assures that the ranges computed include the actual range of the signal. However, non-linear systems require the application of approximations (see eqn. 4.18 and eqn. 4.19) that do not guarantee the inclusion of the actual range, since they lead to over- or underestimation of the range. In these cases, it is a common approach

to apply simulation-based methods to obtain the range of signals [SK95].

The scaling of a signal $s_i \in S$ can be computed by means of:

$$p_i = \left\lfloor \log_2 \left(\max \left\{ |\overline{range_{s_i}}|, |range_{s_i}| \right\} \right) \right\rfloor + 1. \quad (4.21)$$

Here, $range_{s_i}$ represents the interval of possible values of s_i .

Once the scaling is computed, it is fixed to all signals and it is not changed during the word-length selection task.

4.2.3. Error estimation

In this subsection, a fast and accurate noise estimation method is sought.

As aforementioned, the quantization of signal s_i is modeled as the addition of an AWGN noise $n_i[n]$ with mean μ_i and variance σ_i^2 , related to equations 4.1, 4.2. This noise can be modeled using affine forms by means of assigning a uniform distribution to the error intervals. Thus, a uniform distribution with mean μ_i and variance σ_i^2 can be represented with an affine form which possesses only one error term:

$$\hat{x}_{uniform} = -\mu_i + \sqrt{12\sigma_i^2}\epsilon. \quad (4.22)$$

However, in order to estimate the effect of the deviation from the original behavior of an algorithm with feedback loops [Lóp04], it is necessary to generate a new affine form $\hat{x}[n]$ per each simulation time instant n . Thus, the quantization noise produced by signal s_i is modeled as a sequence of affine forms as follows:

$$\hat{n}_i[n] = -\mu_i + \sqrt{12\sigma_i^2}\epsilon_{i,n} = \epsilon'_{i,n} \quad (4.23)$$

Thus, by performing an affine simulation in time, it is possible to know at each moment the origin of a particular error term (i) and the moment when it was generated (n). Note

that it is possible to use a single error term ϵ' which encapsulates the mean value and the error term ϵ , in order to simplify the AA simulations. Once the simulation is finished, the values of μ_i and $\sqrt{12\sigma^2}\epsilon_{i,n}$ can be placed back. Thus, the AA-based simulation can be made independent on the particular statistical parameters of each quantization, which is desirable in order to obtain a parameterizable noise model.

The expression of a given output \hat{Y} of the algorithm with $|S|$ noise sources is

$$\hat{Y}[n] = Y_0[n] + \sum_{i=0}^{|S|-1} \sum_{j=0}^{n-1} A_{i,j}[n] \epsilon'_{i,j} \quad (4.24)$$

where $Y_0[n]$ is the value of the output of the algorithm using real arithmetic and the summation is the contribution of the quantization noise sources. $A_{i,j}[n]$ is a function that depends on the inputs and the constants used in multiplications by a constant.

The error \hat{Err}_Y at the output is

$$\hat{Err}_Y[n] = \sum_{i=0}^{|S|-1} \sum_{j=0}^{n-1} A_{i,j}[n] \epsilon'_{i,j} \quad (4.25)$$

The value of the error comprises a collection of affine forms at each time step n . The power of the output's quantization noise can be approximated by the MSE (Mean Square Error) which is estimated as the mean value of the power of the summations of the uniform distributions at each time step n as in 4.26, 4.27. These equations rely on the fact that each error terms $\epsilon'_{i,n}$ are uncorrelated to each other.

$$\begin{aligned}
P\left(\hat{Err}_Y[n]|_{n=n'}\right) &= Var(\hat{Err}_Y[n']) + \left(E\left[\hat{Err}_Y[n']\right]\right)^2 \\
&= Var\left\{\sum_{i=0}^{|S|-1}\sum_{j=0}^{n'-1}A_{i,j}[n']\epsilon'_{i,j}\right\} + \left(E\left[\sum_{i=0}^{|S|-1}\sum_{j=0}^{n'-1}A_{i,j}[n']\epsilon'_{i,j}\right]\right)^2 \\
&= \sum_{i=0}^{|S|-1}\sum_{j=0}^{n'-1}Var\{A_{i,j}[n']\epsilon'_{i,j}\} + \left(\sum_{i=0}^{|S|-1}\sum_{j=0}^{n'-1}A_{i,j}[n']E[\epsilon'_{i,j}]\right)^2 \\
&= \sum_{i=0}^{|S|-1}\sum_{j=0}^{n'-1}A_{i,j}^2[n']Var\{\epsilon'_{i,j}\} + \left(\sum_{i=0}^{|S|-1}\mu_i\sum_{j=0}^{n'-1}A_{i,j}[n']\right)^2 \\
&= \sum_{i=0}^{|S|-1}\sigma_i^2\sum_{j=0}^{n'-1}A_{i,j}^2 + \left(\sum_{i=0}^{|S|-1}\mu_i\sum_{j=0}^{n'-1}A_{i,j}[n']\right)^2 \tag{4.26}
\end{aligned}$$

$$\begin{aligned}
P\left(\hat{Err}_Y[n]\right) &= \frac{1}{N}\sum_{n'=0}^{N-1}P\left(\hat{Err}_Y[n]|_{n=n'}\right) \\
&= \frac{1}{N}\sum_{n'=0}^{N-1}\left(\sum_{i=0}^{|S|-1}\sigma_i^2\sum_{j=0}^{n'-1}A_{i,j}^2 + \left(\sum_{i=0}^{|S|-1}\mu_i\sum_{j=0}^{n'-1}A_{i,j}[n']\right)^2\right) \tag{4.27}
\end{aligned}$$

Expressions for the mean and variance can be obtained in a similar fashion:

$$\begin{aligned}
\mu_{\hat{Err}_Y} &= \frac{1}{N}\sum_{n'=0}^{N-1}\mu_{\hat{Err}_Y[n]|_{n=n'}} \\
&= \frac{1}{N}\sum_{n'=0}^{N-1}\left(\sum_{i=0}^{|S|-1}\mu_i\sum_{j=0}^{n'-1}A_{i,j}\right) \tag{4.28}
\end{aligned}$$

$$\sigma_{\hat{Err}_Y}^2 = P(\hat{Err}_Y[n]) - \mu_{\hat{Err}_Y}^2 \tag{4.29}$$

These expressions can be applied to DSP algorithms including multiplications using eqn. 4.19. The result is not exact, since eqn. 4.19 is an approximation. However, they should be exact for LTI systems and expressions 4.27-4.29 should match the well known

expressions 4.6-4.8. Equations 4.6-4.8 can be rewritten as follows:

$$\mu_{LTI} = \sum_{i=0}^{|S|-1} \mu_i \cdot G_i(1) \approx \sum_{i=0}^{|S|-1} \mu_i \sum_{n=0}^{N'-1} g_i[n] \quad (4.30)$$

$$\sigma_{LTI}^2 = \sum_{i=0}^{|S|-1} \sigma_i^2 \cdot \frac{1}{2\pi} \int_{-\pi}^{\pi} |G_i(e^{j\Omega})|^2 d\Omega \approx \sum_{i=0}^{|S|-1} \sigma_i^2 \sum_{n=0}^{N'-1} g_s^2[n] \quad (4.31)$$

$$P_{LTI} = \sigma_{LTI}^2 + (\mu_{LTI})^2 \approx \sum_{i=0}^{|S|-1} \sigma_i^2 \sum_{n=0}^{N'-1} g_s^2[n] + \left(\sum_{i=0}^{|S|-1} \mu_i \sum_{n=0}^{N'-1} g_i[n] \right)^2 \quad (4.32)$$

The LTI system is supposed to be causal ($\forall n < 0, g_i[n] = 0$) and stable ($g_i[n]|_{\rightarrow\infty} = 0$), and N' is chosen such that $\forall n > N', g_i[n] \approx 0$.

In LTI systems the coefficients $A_{i,j}[n]$ multiplying each $\epsilon'_{i,j}$ depend only on $g_i[n]$ and are equal to

$$A_{i,j}^{LTI}[n] = \begin{cases} g_i[n-j] & \text{if } n > 0 \\ 0 & \text{otherwise} \end{cases} \quad (4.33)$$

Equation 4.27 turns into:

$$P_{\hat{ErrorY}}^{LTI} = \frac{1}{N} \sum_{n=0}^{N-1} \left(\sum_{i=0}^{|S|-1} \sigma_i^2 \sum_{j=0}^{n-1} g_i^2[n-j] + \left(\sum_{i=0}^{|S|-1} \mu_i \sum_{j=0}^{n-1} g_i[n-j] \right)^2 \right) \quad (4.34)$$

Note that equations 4.30-4.32 assume that the LTI system is in steady state, thus the transient must be removed from the computation of the MSE. Hereby, equation 4.34 only matches 4.32 if the affine simulation is performed during M iterations ($M \gg N + N'$), where N' is such that $\forall n > N', g_i[n] \approx 0$ and the first K iterations ($K > N'$) are removed

from the computation. Thus

$$\begin{aligned}
P_{\hat{E}_{err_Y}}^{LTI} &= \frac{1}{M-K} \sum_{n=K}^{M-1} \left(\sum_{i=0}^{|S|-1} \sigma_i^2 \sum_{j=0}^{n-1} g_i^2[n-j] + \left(\sum_{i=0}^{|S|-1} \mu_i \sum_{j=0}^{n-1} g_i[n-j] \right)^2 \right) \\
&\approx \frac{1}{M-K} \sum_{n=K}^{M-1} \left(\sum_{i=0}^{|S|-1} \sigma_i^2 \sum_{j=0}^{N'} g_i^2[n-j] + \left(\sum_{i=0}^{|S|-1} \mu_i \sum_{j=0}^{N'} g_i[j] \right)^2 \right) \\
&= \frac{1}{M-K} (M-K) \cdot \left(\sum_{i=0}^{|S|-1} \sigma_i^2 \sum_{j=0}^{N'} g_i^2[j] + \left(\sum_{i=0}^{|S|-1} \mu_i \sum_{j=0}^{N'} g_i[j] \right)^2 \right) \\
&= \sum_{i=0}^{|S|-1} \sigma_i^2 \sum_{j=0}^{N'} g_i^2[j] + \left(\sum_{i=0}^{|S|-1} \mu_i \sum_{j=0}^{N'} g_i[j] \right)^2 \\
&= P^{LTI}.
\end{aligned} \tag{4.35}$$

Similarly, equations 4.28 and 4.29 can be matched to equations 4.30 and 4.31, respectively, which validates the approach for LTI algorithms.

However, for LTI systems in steady state it is possible to simplify the noise estimation by means of modifying the expression of the noise terms such as:

$$\hat{n}_s[n] = \begin{cases} -\mu_s + \sqrt{12\sigma^2} \epsilon_{s,n} = \epsilon'_{s,n} & \text{if } n = 0 \\ 0 & \text{otherwise} \end{cases} \tag{4.36}$$

It can be inferred that

$$A_{i,0}[n] = g_i[n]. \tag{4.37}$$

Therefore it is possible to rewrite the set of eqns. 4.11-4.12 in order to relate them to the amplitudes of the error terms at the output of the system $\hat{Y}[n]$ as shown in eqns. 4.38-4.41.

$$\mathbf{v}^{LTI} = \left\langle \sum_{j=0}^{M-1} Ay_{0,0}^2[j], \dots, \sum_{j=0}^{M-1} Ay_{|S|-1,0}^2[j] \right\rangle \quad (4.38)$$

$$\mathbf{m}^{LTI} \equiv \left\langle \sum_{j=0}^{M-1} A_{0,0}[j], \dots, \sum_{j=0}^{M-1} A_{|S|-1,0}[j] \right\rangle \quad (4.39)$$

$$\mathbf{M}^F \equiv \begin{bmatrix} m_{0,0} & \cdots & m_{|S|-1,0} \\ & \ddots & \\ m_{0,|S|-1} & \cdots & m_{|S|-1,|S|-1} \end{bmatrix} \quad (4.40)$$

$$m_{i_1, i_2} = \sum_{n=0}^{M-1} \left(A_{i_1,0}[n] \sum_{i_2=0}^{n-1} A_{i_2,0}[n] \right) \quad (4.41)$$

Matrix \mathbf{M}^F is introduced (eqn.4.40) if there are forks and their outputs are quantized. As aforementioned, the noise parameters of the fork outputs vary with the word-length arrangement due to the current fixed-point parameters of these signals. Hence, the necessity to modify the elements in \mathbf{v} and \mathbf{m} related to these. Algorithm 4.1 shows how to *update* vectors \mathbf{v} and \mathbf{m} for a given conditioned graph G_{SFG} . This algorithm outputs the new parameterization vectors as if the outputs of forks would have been arranged following the cascade model during the AA simulation. Once \mathbf{v}^{LTI} and \mathbf{m}^{LTI} are updated, equations 4.6-4.8 can be applied.

For non-linear systems, or LTI systems with transient, the power of the output noise (eqn. 4.27), as well as the mean and the variance, can be expressed by means of vectors \mathbf{v}^{NL} , \mathbf{m}^{NL} , and matrix \mathbf{M}^{NL} as shown in equations (eqns. 4.42-4.46). Now M represents the number of samples of the inputs used to estimate the output noise. Algorithm 4.2 shows how to update \mathbf{m}^{NL} , \mathbf{v}^{NL} and \mathbf{M}^{NL} if forks are quantized.

$$P_o = \frac{1}{M} \left(\sigma^2 \cdot \mathbf{v}^{NL^T} + \boldsymbol{\mu} \cdot \mathbf{M}^{NL} \boldsymbol{\mu}^T \right) \quad (4.42)$$

$$\mu_o = \frac{1}{M} \left(\boldsymbol{\mu} \cdot \mathbf{m}^{NL^T} \right) \quad (4.43)$$

$$\sigma_o^2 = P_o - \mu_o^2 \quad (4.44)$$

$$\mathbf{v}^{NL} \equiv \left\langle \sum_{n=0}^{M-1} \sum_{j=0}^{n-1} Ay_{0,j}^2[n], \dots, \sum_{n=0}^{M-1} \sum_{j=0}^{n-1} Ay_{|S|-1,j}^2[n] \right\rangle \quad (4.45)$$

$$\mathbf{m}^{NL} \equiv \left\langle \sum_{n=0}^{M-1} \sum_{j=0}^{n-1} A_{0,j}[n], \dots, \sum_{n=0}^{M-1} \sum_{j=0}^{n-1} A_{|S|-1,j}[n] \right\rangle \quad (4.46)$$

$$\mathbf{M}^{NL} \equiv \begin{bmatrix} m_{0,0} & \cdots & m_{|S|-1,0} \\ & \ddots & \\ m_{0,|S|-1} & \cdots & m_{|S|-1,|S|-1} \end{bmatrix} \quad (4.47)$$

$$m_{i_1, i_2} = \sum_{n=0}^{M-1} \left(\sum_{j_1=0}^{n-1} A_{i_1, j_1}[n] \sum_{j_2=0}^{n-1} A_{i_2, j_2}[n] \right) \quad (4.48)$$

Algorithm 4.1 Parameter updating due to forks for LTI systems

Input: $G_{SFG}(V, S)$, \mathbf{v} , \mathbf{m} , \mathbf{M}^F

Output: \mathbf{v}_{new} , \mathbf{m}_{new}

- 1: **for all** $v_f \in V_F$ **do**
 - 2: obtain the ascendant sorting function $\alpha_{v_f} : \mathbb{N} \rightarrow S$ of $out(v_f)$
 - 3: **for** $i=0$ to $|out(v_f)| - 1$ **do**
 - 4: $\mathbf{m}_{new}[\alpha_{v_f}(i)] = \sum_{j=i}^{|out(v_f)|-1} \mathbf{m}[\alpha_{v_f}(j)]$
 - 5: $\mathbf{v}_{new}[\alpha_{v_f}(i)] = \sum_{j=i}^{|out(v_f)|-1} \mathbf{v}[\alpha_{v_f}(j)]$
 $\quad \quad \quad + 2 \cdot \sum_{k=i}^{|out(v_f)|-1} \sum_{l=k}^{|out(v_f)|-1} \mathbf{M}^F[\alpha_{v_f}[k], \alpha_{v_f}[l]]$
 - 6: **end for**
 - 7: **end for**
-

Algorithm 4.2 Parameter updating due to forks for LTI and non-linear systems

Input: $G_{SFG}(V, S), v, m, M, M$ **Output:** v_{new}, m_{new}

```
1: for all  $v_f \in V_f$  do
2:   obtain the ascendant sorting function  $\alpha_{v_f} : N \rightarrow S$  of  $out(v_f)$ 
3:   for  $i=0$  to  $|out(v_f)| - 1$  do
4:      $m_{new}[\alpha_{v_f}(i)] = \sum_{j=i}^{|out(v_f)|-1} m[\alpha_{v_f}(j)]$ 
5:      $v_{new}[\alpha_{v_f}(i)] = \sum_{j=i}^{|out(v_f)|-1} v[\alpha_{v_f}(j)] + \sum_{k=i}^{|out(v_f)|-1} \sum_{l=i}^{|out(v_f)|-1} M[\alpha_{v_f}(k), \alpha_{v_f}(l)]$ 
6:   end for
7: end for
8:  $S_{Fout} = s \in out(V_F)$ 
9:  $S_F = s \in in(V_F) \cup S_{Fout}$ 
10:  $S' = S \setminus S_F$ 
11: for all  $v_{f1} \in V_F$  do
12:   for  $i=0$  to  $|out(v_{f1})| - 1$  do
13:     for all  $s_j \in S'$  do
14:        $M_{new}[i, j] = M_{new}[j, i] = \sum_{k=i}^{|out(v_{f1})|-1} M[\alpha_{v_{f1}}(k), i]$ 
15:     end for
16:     for all  $v_{f2} \in V_F$  do
17:       for  $j=0$  to  $|out(v_{f2})| - 1$  do
18:          $M_{new}[\alpha_{v_{f1}}(i), \alpha_{v_{f2}}(j)] = \sum_{k=i}^{|out(v_{f1})|-1} \sum_{l=j}^{|out(v_{f2})|-1} M[\alpha_{v_{f1}}(k), \alpha_{v_{f2}}(l)]$ 
19:          $M_{new}[\alpha_{v_{f2}}(j), \alpha_{v_{f1}}(i)] = M_{new}[\alpha_{v_{f1}}(i), \alpha_{v_{f2}}(j)]$ 
20:       end for
21:     end for
22:   end for
23: end for
```

4.3. Word-Length selection

In this subsection four representative heuristic algorithms to perform word-length selection are presented:

- Uniform word-length selection (*UWL*) (alg. 4.3)
- Gradient-descent word-length selection (*GRAD*) (alg. 4.4)
- Modified gradient-descent word-length selection (*GRAD*₂) (alg. 4.5)
- Simulated annealing based word-length selection (*SA*)

The algorithms have been ascendantly sorted in terms of complexity and optimality and all fit the scheme presented in fig. 4.1.

The first one, *UWL*, is shown in algorithm 4.3 and it represents the original approach to word-length selection inherited from a microprocessor's point of view. The input to the algorithm is graph G_{SFG} with undefined couples (p, n) and an error constraint E_{max} that can represent power or a variance of the quantization output noise. The output is graph G_{SFG} with all fixed-point parameters (p, n) defined. The optimization finds the minimum value for both p and n that complies with the noise constraints when applied to all signals in G_{SFG} . Yet a very fast optimization procedure, the results produced in terms of area are quite far from the optimal. It is common to use it as a starting point to more complex optimization procedures, in fact, the other three optimization approaches have as a starting point a modified version of this procedure.

Word-length selection *GRAD* is developed in alg. 4.4. The optimization procedure starts with a rough solution based on a modification of *UWL* that meets the noise constraint (see lines 1 and 2). Then the word-lengths of signals are decreased sequentially selecting the signal that produces the steepest descent of the cost function. This approach has been previously applied in [CSPL01]. Since word-length selection is normally a task prior to any

Algorithm 4.3 Uniform word-length optimization (*UWL*)

Input: $G_{SFG}(V, S)$ with no fixed-point information, error constraint E_{max}

Output: word-length optimized $G_{SFG}(V, S)$

- 1: Perform *scaling* and initialize $P = \{p_0, \dots, p_{|S|-1}\}$
 - 2: $p_{max} = \max(p_i \in P)$
 - 3: $\forall p_i \in P, p_i = p_{max}$
 - 4: find minimum n that complies with noise constraint E_{max}
when $\forall n_i \in N = \{n_0, \dots, n_{|S|-1}\}, n_i = n$
-

architectural or high-level synthesis there is no information regarding the final architecture of the system so it is common to assume a direct implementation of the system (i.e. there is an available resource per each arithmetic operation). The procedure has little computational complexity and it enables to obtain quasi-optimal solutions. However, since it is based on local information, one of its main drawbacks is that the solution can get stuck in a local minimum.

A modification of the gradient-descent algorithm ($GRAD_2$) was presented in [CCL03a] and it is shown in alg. 4.5. Here, the decision of whether a signal's word-length must be decreased one unit or not is based on a deeper cost function exploration. For a given word-length set the next candidate to be decreased is the one which most reduces the cost when reduced to the minimum value that the noise constraint allows. The computational complexity is increased, but the results have a better chance to be closer to the global minimum than with the non-modified gradient-descent algorithm.

The last word-length selection procedure, SA , is based on simulated annealing. Basically, it follows the scheme proposed in alg. 3.1 with a few modifications. First, the mapping function m is now very basic because the architecture of the system is fixed and with a one-to-one relationship between operation and resources, where one resource is allocated for each operation. The other modification is that the computation of the area is also straightforward and it does not correspond to alg. 3.2. The area is computed by traversing the available

Algorithm 4.4 Gradient-descent optimization (*GRAD*)

Input: $G_{SFG}(V, S)$ with no fixed-point information, error constraint E_{max}

Output: word-length optimized $G_{SFG}(V, S)$

- 1: Perform *scaling* and initialize $P = \{p_0, \dots, p_{|S|-1}\}$
 - 2: find minimum n that complies with noise constraint E_{max}
when $\forall n_i \in N = \{n_0, \dots, n_{|S|-1}\}, n_i = n$

 - 3: Compute output error E
 - 4: $A_{min} = \infty$
 - 5: **repeat**
 - 6: $s_{candidate} = undefined$
 - 7: **for all** $s_i \in S$ **do**
 - 8: $n_i = n_i - 1$
 - 9: Compute output error E
 - 10: **if** $E < E_{max}$ **then**
 - 11: Compute area A
 - 12: **if** $A < A_{min}$ **then**
 - 13: $A_{min} = A$
 $s_{candidate} = s_i$
 - 14: **end if**
 - 15: **end if**
 - 16: restore n_i
 - 17: **end for**
 - 18: **if** $s_{candidate}^\neg = undefined$ **then**
 - 19: $n_{candidate} = n_{candidate} - 1$
 - 20: **end if**
 - 21: **until** $s_{candidate}^\neg = undefined$
-

Algorithm 4.5 Gradient-descent optimization ($GRAD_2$) [CCL03a]

Input: $G_{SFG}(V, S)$ with no fixed-point information, error constraint E_{max}

Output: word-length optimized $G_{SFG}(V, S)$

- 1: Perform *scaling* and initialize $P = \{p_0, \dots, p_{|S|-1}\}$
 - 2: find minimum n that complies with noise constraint E_{max}
 when $\forall n_i \in N = \{n_0, \dots, n_{|S|-1}\}, n_i = n$

 - 3: Compute output error E
 - 4: $A_{min} = \infty$
 - 5: **repeat**
 - 6: $s_{candidate} = \text{undefined}$
 - 7: **for all** $s_i \in S$ **do**
 - 8: $n_{tmp} = n_i$
 - 9: find minimum n_i that complies with noise constraint E_{max}
 - 10: **if** $n_i \bar{\cap} = n_{tmp}$ **then**
 - 11: Compute area A
 - 12: **if** $A < A_{min}$ **then**
 - 13: $A_{min} = A$
 - 14: $s_{candidate} = s_i$
 - 15: **end if**
 - 16: **end if**
 - 17: restore n_i
 - 18: **end for**
 - 19: **if** $s_{candidate} \bar{\cap} = \text{undefined}$ **then**
 - 20: $n_{candidate} = n_{candidate} - 1$
 - 21: **end if**
 - 22: **until** $s_{candidate} \bar{\cap} = \text{undefined}$
-

resource list and adding up the areas of the resources, taking into account the word-lengths of the input and output signals of the associated operation. Finally, the movements are not applied to the mapping, since this is fixed, but to the word-length of signals from graph G_{SFG} . The movements are as follows:

- Increase the value of n of random $s_i \in S \setminus in(V_F) \cup out(V_D)$.
- Decrease the value of n of a random signal s_i .

The second movement was given a double probability of occurrence, since this sped up the optimization process. It is well known that SA is characterized by very long convergence times, although it produces quasi-optimal results.

In the results section, the four optimization methods are compared for non-shared resources. Also, the impact of performing WLA before HLS is analyzed.

4.4. Results

The results are divided into two main sections:

- i) The accuracy and computational performance of the AA-based noise estimation method presented in subsection 4.2.3 are evaluated. Since the noise estimator is conceived to be integrated within a combined WLA and HLS synthesizer, it is very important to verify if there are any significant time improvements in comparison to the traditional simulation-based option. Also, the accuracy must be high to assure the correct synthesis of DSP circuits.
- ii) A comparison of the four word-length selection methods presented in 4.3 is performed. First, the impact of using the different methods to non-sharing architectures is analyzed. Then, the impact on HLS of a prior WLA task is determined for both homogeneous and heterogeneous architectures. The purpose of these tests is to show the necessity of

integrating WLA and HLS into a single task. The interdependencies between these two tasks are sought to be exposed with these experiments.

4.4.1. Error estimation results

The benchmarks used to test the error estimation technique are: (i) an ITU RGB to YCrCb converter (*ITU*) [CCL03a]; (ii) a 4th-order IIR filter (*IIR*₄) [KKS00]; (iv) a 3x3 vector scalar multiplication (*VEC*_{3x3}); (v) a mean power 1st-order IIR filter (*POW*); (vi) a 5th-order LMS filter (*LMS*₅) [Hay02]. Note that the first two benchmarks are LTI, while the remaining are non-linear. All benchmarks are fed with 12-bit inputs and 12-bit constants and the noise constraint is given as an SQNR ranging from 4 dB to 120 dB. The *LMS*₅ benchmark had as reference signal a synthetic random phase tone, and as received signal the reference with five echoes added. The remaining benchmarks have uniformly distributed noises as inputs. The procedure to carry out the test is as follows:

- Extract noise parameters: eqns. 4.11-4.40 for LTI or eqns. 4.45-4.47 for non-linear.
- Perform gradient-descent word-length selection using a variance or power noise estimator: eqns. 4.6 and 4.8 for LTI systems, and eqns. 4.44 and 4.42 for non-linear.
- Perform a fixed-point bit-true simulation and use it as reference.
- Compare the estimate and the reference.

Table 4.1 contains some results for *LMS*₅. The first column shows the SQNR constraint used during the optimization. The second column contains the relative error obtained by the estimator using a fixed-point simulation as a reference. The third column indicates the number of iterations (i.e. number of quantization noise estimates) required during the gradient-descent optimization. The computation times required to perform the word-length selection

Table 4.1: Error estimation results for 5th-order LMS filter (LMS_5)

SQNR (dB)	Pow. Est. error (%)	Iter.	Est.-based Opt. (sec.)	FXP Opt.* (sec.)	Sim.-based Opt. Δ (sec.)	Speedup (times)
120	0.73	7105	21.17	0.44	3104.89	$\times 146.65$
110	0.69	6687	19.828	0.438	2928.91	$\times 147.72$
100	0.56	5889	17.343	0.453	2667.72	$\times 153.82$
90	0.69	7067	21.203	0.453	3201.35	$\times 150.99$
80	0.83	6649	19.89	0.453	3012	$\times 151.43$
70	0.22	5813	17.266	0.437	2540.28	$\times 147.13$
60	1.58	7219	20.125	0.453	3270.21	$\times 162.50$
50	2.14	6763	14.938	0.453	3063.64	$\times 205.09$
40	3.63	5813	12.656	0.453	2633.29	$\times 208.07$
35	0.29	4787	9.891	0.453	2168.51	$\times 219.24$
30	51.34	7333	16.266	0.453	3321.85	$\times 204.22$
25	54.07	7067	15.625	0.469	3314.42	$\times 212.12$
20	82.70	6725	14.859	0.453	3046.43	$\times 205.02$
16	72.91	5699	12.375	0.453	2581.65	$\times 208.62$
12	162.97	7333	16.25	0.453	3321.85	$\times 204.42$
8	133.30	6649	14.718	0.438	2912.26	$\times 197.87$
4	82.27	5471	10.641	0.437	2390.83	$\times 224.68$
* a single FXP simulation						
Δ FXP sim. time \times Iterations						

using the proposed estimation (eqn. 4.8 or 4.42) are in the fourth column. The fifth column shows the time required to perform the fixed-point simulation used as a reference to assess the accuracy of the estimation. The next column contains an estimation of the time that would have been required to perform a simulation-based gradient-descent word-length selection (time of a single bit-true simulation \times no. of optimization iterations). Finally, the last column contains the computation time speedup obtained due to the use of the novel estimation method proposed.

The accuracy results (second column) yield that the estimation provides highly acceptable results within the range 120-40 dB, that varies from 0.22% to 3.63%, while it degrades

significantly for SQNR values smaller than 40 dB. However, the range 120-40 dB meets the requirements that most DSP algorithms require. The estimation model relies on the fact that the quantization noise produced by each signal is much smaller than its range. Therefore, for low values of SQNR this assumption does not remain valid. For this particular example, the point that divides *acceptable* results from *non-acceptable* results can be located at 40 dB. Moreover, the estimation is based on the assumption that second order noise effects are negligible. This is not true if there exists any non-linearity in the algorithm, and it gets worse in the presence of feedbacks. LMS_5 contains both, and even so, the accuracy performance meets the standard.

Both the extraction of the noise parameters and the fixed-point simulation are fed with 5000-sample input vectors. The time required to perform the parameters extraction is around 30 minutes. This computation time might seem quite large when compared to the time required to perform a simulation-based optimization (see 6th column), however, the main point of the proposed estimation technique is that it enables to perform as many optimizations as required in very short times. For instance, it can be seen how the optimization times using the estimator are around a few seconds (column 5) while a simulation-based optimization requires times two order of magnitude larger (column 7). The computation time speedup ranges from $\times 146$ to $\times 224$ which proves the strength of the proposed estimation method.

The accuracy obtained by means of a gradient-descent optimization under different SQNR constraints for the different benchmarks is presented in table 4.2. The first column indicates the benchmark used. The second one, the SQNR range. Then, the third column shows the minimum, maximum and mean values of the relative error of the noise estimation when compared to a fixed-point simulation. Note that the SQNR has been divided into two intervals: [120,40], which is desired to have high accuracy, and (40,3], where a low accuracy is expected. The mean error in the estimation in the SQNR interval [120,40] ranges from 0.55% to 3.64% which corroborates the high accuracy of the estimation method proposed.

Table 4.2: Accuracy of the estimation method.

Alg.	SQNR range	Pow. est. err. (%)		
		min.	max.	mean.
<i>ITU</i>	[120,40]	0.09	1.85	0.55
	(40,3]	0.48	4.23	1.21
<i>IIR</i> ₄	[120,40]	0.001	1.13	0.94
	(40,3]	0.04	3.56	0.85
<i>VEC</i> _{3×3}	[120,40]	0.003	2.59	0.70
	(40,3]	1.51	32.58	10.38
<i>POW</i>	[120,40]	0.02	64.20	3.64
	(40,3]	0.18	79.76	26.28
<i>LMS</i> ₅	[120,40]	0.04	6.13	1.16
	(40,3]	0.29	209.26	74.34

It is worth mentioning that the LTI algorithms as well as the non-feedback non-linear algorithms perform very well for the whole range [120,3], while the non-linear algorithms with feedbacks produce poor results for the range [40,3], as expected.

Table 4.3 contains results on computational performance. The computer used to carry out the experiments has a 1.66 GHz Intel Core Duo processor and 1 GB of RAM. The first column indicates the name of the benchmark. The second column holds the number of samples used to perform the bit-true simulations, and the next column indicates the number of samples used to extract the noise parameters. The fourth column has the number of iterations required to perform a gradient-descent word-length selection. Column 5 contains the noise parameterization time. The next column shows the minimum, maximum and mean values of the computation time required to perform a gradient-descent optimization using the estimator proposed. The seventh column contains the minimum, maximum and mean values of the speedup obtained comparing to a simulation-based approach.

Regarding noise parameterization time in column 4, the results show that LTI systems require a very short time, while this time is increased for non-linear systems. Moreover, the presence of feedbacks in non-linear systems increased the computation time notably.

Table 4.3: Computational performance of noise estimation method.

Alg.	Sim. samples	Par. samples	Par. time (sec.)	No. iterations (mean)	Est. time (sec)			Opt. speedup (times)		
					min.	max.	mean.	min.	max.	mean.
<i>ITU</i>	10000	1	0.02	119.42	0.02	0.06	0.03	×395	×1210	×708
<i>IIR₄</i>	10000	5000	6.89	283.11	0.03	0.16	0.10	×711	×1140	×871
<i>VEC_{3×3}</i>	10000	10000	5.72	155.24	0.02	0.13	0.07	×96	×573	×378
<i>POW</i>	10000	10000	96.61	33.83	0.02	0.03	0.03	×27	×215	×99
<i>LMS₅</i>	5000	5000	1980.20	6273.19	8.52	21.34	16.33	×146	×231	×177

However, even for the worst case (*LMS₅*), the time speedups gained (last column) in contrast to a simulation-based approach are outstanding (up to a 1210-fold). The estimation-based optimization time (column 6) ranges from 0.02 to 21.34 secs, which is a very good result taking into account that hundreds of estimates, even thousands, are performed during the optimization. When compared to a pure simulation-based approach the optimization speedup ranges from x27 to x1210. Thus, the computation performance of the method exposed is clearly advantageous.

Summarizing, the proposed noise estimation method produces fast (up to a x1210) and accurate estimates (average error around 4% for noise constraints within [120,40] dB) that enable the application of powerful word-length optimization methods, and therefore, a significant reduction in implementation costs.

4.4.2. Word-length selection techniques comparison

The same benchmarks used in chapter 3 (*ITU*, *LAT₃*, *IIR₄* and *FIR₈*) are tested in this subsection to perform a comparison between the different word-length selection techniques presented in subsection 4.4.2. All algorithms are assigned 8-bit inputs and 12-bit constant coefficients. The algorithms have been tested under different latency and output noise scenarios assuming a system clock of 125 MHz, as explained in section 3.4. The target devices belong to the Xilinx Virtex-II family, being the area results normalized with respect to the

Table 4.4: *UWL* vs. *GRAD*, *GRAD*₂, *SA*: direct homogeneous implementations.

Bench.	σ^2	<i>GRAD</i> (%)	<i>GRAD</i> ₂ (%)	<i>SA</i> (%)
<i>ITU</i>	10^{-1}	63.97	63.97 / 0.00*	65.33 / 3.78
	10^{-2}	63.41	63.41 / 0.00	64.08 / 1.83
	10^{-3}	63.03	63.03 / 0.00	64.69 / 4.51
<i>LAT</i> ₃	10^{-3}	71.66	72.91 / 4.39	72.97 / 4.60
	10^{-4}	63.84	69.65 / 16.37	69.76 / 16.37
	10^{-5}	61.07	64.63 / 9.13	64.63 / 9.13
<i>IIR</i> ₄	10^{-3}	64.96	67.11 / 6.16	68.67 / 10.61
	10^{-4}	62.31	62.02 / -0.77	64.07 / 4.67
	10^{-5}	54.63	56.66 / 4.46	58.86 / 9.32
<i>FIR</i> ₈	10^{-3}	45.70	61.10 / 28.36	61.56 / 29.21
	10^{-4}	44.09	42.25 / -3.29	48.85 / 8.51
	10^{-5}	33.64	36.11 / 3.73	40.87 / 10.90
<i>All</i>		57.69	60.24 / 5.69	62.03 / 9.45
* with respect to <i>UWL</i> / with respect to <i>GRAD</i>				

XC2V40 device. The different implementations are performed following the classical approach: first, WLA is carried out, and then HLS. WLA is performed by applying the four techniques from section 4.3 (*UWL*, *GRAD*, *GRAD*₂ and *SA*), always assuming a direct (non-sharing) homogeneous implementation. HLS is carried out always after WLA, for direct implementations (non-sharing architectures) and for resource-sharing architectures. Direct implementations require a straight-forward HLS algorithm since there is a one-to-one relationship between operations and resources. Resource-sharing HLS was explained in 3.3. Both WLA SA-based and HLS SA-based are repeated four times and the best results are selected (as explained in 3.4).

Direct implementations

Direct implementations are those with no resource-sharing, thus, with a one-to-one relationship between operations and FUs. Table 4.4 presents the area improvements obtained by

GRAD, *GRAD*₂ and *SA* compared to *UWL* for direct implementations using homogeneous-architecture FPGAs. The first column indicates the benchmark used. The next column shows the noise variance constraint used. And the last three columns show the area improvements obtained by *GRAD*, *GRAD*₂ and *SA* word-length selection approaches. Note that *GRAD*₂ and *SA* are also compared to *GRAD*.

The results show that *SA* performs the best, as expected, since it carries out a deeper optimization. In the majority of cases, *GRAD*₂ performs better than *GRAD*, as expected, although there are a couple of experiments (*IIR*₄, $\sigma^2 = 10^{-4}$, *FIR*₈, $\sigma^2 = 10^{-4}$) where *GRAD* performs better than *GRAD*₂. The results corroborate that an MWL approach (i.e. *GRAD*, *GRAD*₂, *SA*) obtained much better results than the classical UWL approach (i.e. *UWL*). The area improvements range from 33.64% to 72.97%. Moreover, the benefits obtained by complex word-length selection techniques, such as *SA* and *GRAD*₂, in comparison with the descent gradient word-length selection *GRAD* are significant. *GRAD*₂ improves the results of *GRAD* an average value of 5.69%, while *SA* 9.45%.

The set of word-lengths obtained from these experiments is used to synthesize direct heterogeneous implementations. Now, the mapping between operations and resources is not a one-to-one function, since multiplications can be mapped to LUT-based or embedded multipliers. The mapping has been performed seeking minimum area (+-norm), following a modified version of the mapping algorithm presented in [LVSB05]. Table 4.5 shows the area improvements obtained with respect to *UWL*. The first two columns indicate the name of the benchmark and the noise constraint, respectively. The next column shows the area improvement for *GRAD* compared to *UWL* split in three components: the *norm-inf* improvement, the improvement of LUT-based resources, and the improvement of embedded resources. The last two columns contain similar information about *GRAD*₂ and *SA*. Note that the ∞ -norm improvements are also compared to *GRAD* for *GRAD*₂ and *SA*.

The results yield that the MWL techniques produce significant area improvements with

Table 4.5: *UWL* vs. *GRAD*, *GRAD*₂, *SA*: direct heterogeneous implementations.

Bench.	σ^2	<i>GRAD</i> (%)			<i>GRAD</i> ₂ (%)			<i>SA</i> (%)		
		$\ \mathbf{A}\ _\infty$	A_{LUT}	A_{EMB}	$\ \mathbf{A}\ _\infty$	A_{LUT}	A_{EMB}	$\ \mathbf{A}\ _\infty$	A_{LUT}	A_{EMB}
<i>ITU</i>	10^{-1}	65.21	68.98	50.00	65.21 / 0.00*	69.92	50.00	65.21 / 0.00	70.73	50.00
	10^{-2}	66.67	64.85	66.67	66.67 / 0.00	64.85	66.67	66.67 / 0.00	64.85	66.67
	10^{-3}	66.65	64.92	66.67	66.65 / 0.00	64.92	66.67	66.67 / 0.05	65.60	66.67
<i>LAT</i> ₃	10^{-3}	70.01	70.01	75.00	71.74 / 5.76	71.4	75.00	71.99 / 6.60	71.99	75.00
	10^{-4}	65.21	61.53	70.00	70.00 / 13.76	68.90	70.00	70.00 / 13.76	69.08	70.00
	10^{-5}	62.25	59.56	70.00	65.93 / 9.74	63.50	70.00	65.93 / 9.74	63.50	70.00
<i>IIR</i> ₄	10^{-3}	63.95	63.95	66.67	68.20 / 11.77	68.20	66.67	68.89 / 13.68	68.89	66.67
	10^{-4}	62.50	63.69	62.50	62.50 / 0.00	65.41	62.50	62.54 / 0.10	58.01	75.00
	10^{-5}	58.52	55.06	62.50	62.50 / 9.60	60.23	62.50	62.50 / 9.60	59.52	62.50
<i>FIR</i> ₈	10^{-3}	36.00	36.00	50.00	54.76 / 29.32	54.76	50.00	55.39 / 30.29	55.39	50.00
	10^{-4}	36.48	36.48	50.00	33.99 / -3.92	33.99	50.00	39.23 / 4.34	39.23	50.00
	10^{-5}	33.33	21.62	33.33	33.33 / 0.00	21.62	33.33	33.33 / 0.00	27.42	33.33
<i>All</i>		57.23	55.55	60.28	60.12 / 6.34	59.00	60.28	60.69 / 7.60	59.52	61.32
* with respect to <i>UWL</i> / with respect to <i>GRAD</i>										

respect to an *UWL* approach. The improvements range from 42.25% to 69.76%. *SA* is the technique that provides better results, followed by *GRAD*₂ and *GRAD*. *SA* performs 7.60% better than *GRAD*, and *GRAD*₂ 6.34%, which indicates that deeper optimization methods achieve better results. Only in a experiment (*FIR*₈, $\sigma^2 = 10^{-4}$), *GRAD* performs better than *GRAD*₂.

It is interesting to remark that even though the overall results indicate that *SA* performs the best, followed by *GRAD*₂ and *GRAD*, as did happen in the homogeneous case, a closer look to the table arises mismatches between the expected behavior of these three word-length selection techniques. For instance, *ITU*, $\sigma^2 = 10^{-1}$, had *SA* performing better than *GRAD*₂ for homogeneous implementations, while they perform alike for heterogeneous implementations. In total, five out of the total twelve constraint scenarios present mismatches with respect to the homogeneous case. This shows how the lack of architectural information during *WLA* may lead to inconsistencies in the final results, in this case due to the lack of

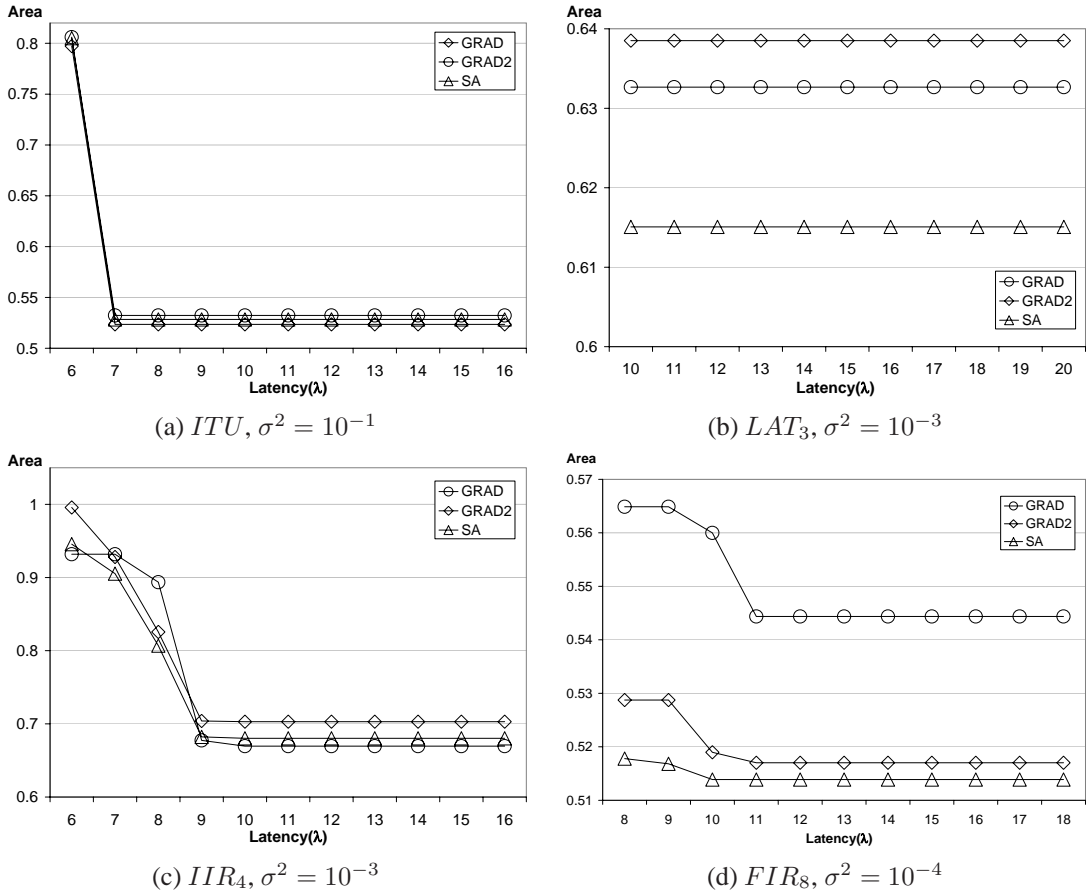


Figure 4.4: Resource-sharing HLS: Homogeneous architectures

information about embedded resources.

Resource-sharing implementations

Fig. 4.4 depicts some area vs. latency results on the comparison of SA , $GRAD_2$ and $GRAD$ when resource sharing is allowed for homogeneous-resource implementations. The latency ranges from $\lambda_{min}^{MWL-HOM}$ to $\lambda_{min}^{MWL-HOM} + 10$. Fig. 4.4-a shows the implementations results for ITU . The differences between the three optimization methods are minimal, however $GRAD$ performs better than the others, presenting a mismatch with the expected behavior from the direct implementation results, where SA performed the best. There are also mismatches with the direct implementations expected behavior in figures 4.4-b,-c. These

Table 4.6: *GRAD* vs. *GRAD*₂, *SA*: resource-sharing homogeneous architectures.

Bench.	σ^2	Area improvement (%)	
		<i>GRAD</i> ₂ *	<i>SA</i> *
<i>ITU</i>	10^{-1}	-1.63	-0.94
	10^{-2}	0.00	4.60
	10^{-3}	0.00	1.67
<i>LAT</i> ₃	10^{-3}	-0.93	2.78
	10^{-4}	8.50	10.77
	10^{-5}	3.18	3.18
<i>IIR</i> ₄	10^{-3}	-3.42	-0.08
	10^{-4}	-5.02	-0.86
	10^{-5}	3.46	13.54
<i>FIR</i> ₈	10^{-3}	22.82	27.02
	10^{-4}	5.48	6.35
	10^{-5}	1.39	1.58
<i>All</i>		2.72	5.18
* with respect to <i>GRAD</i>			

are due to the fact that the WLA optimizations were intended for direct implementations, therefore, without considering registers, multiplexers, or resource sharing.

This analysis is further completed with table 4.6, which contains implementation results of all benchmarks considering nine different quantization scenarios. A quantization scenario is defined by a couple composed of a word-length selection technique (e.g. *GRAD*, *GRAD*₂, *SA*) and a noise variance constraint. For each quantization scenario, after obtaining the set of fixed-point formats assuming a direct implementation, HLS is performed with latencies ranging from $\lambda_{min}^{MWL-HOM}$ to $\lambda_{min}^{MWL-HOM} + 10$. The first column in the table contains the name of the benchmark. The second, the output noise variance applied. The third column holds the average improvement obtained by *GRAD*₂ with respect to *GRAD*. And the last column the improvement obtained by *SA*. The last row contains the total mean improvements.

In general terms, both techniques perform better than *GRAD*: *GRAD*₂ achieves an

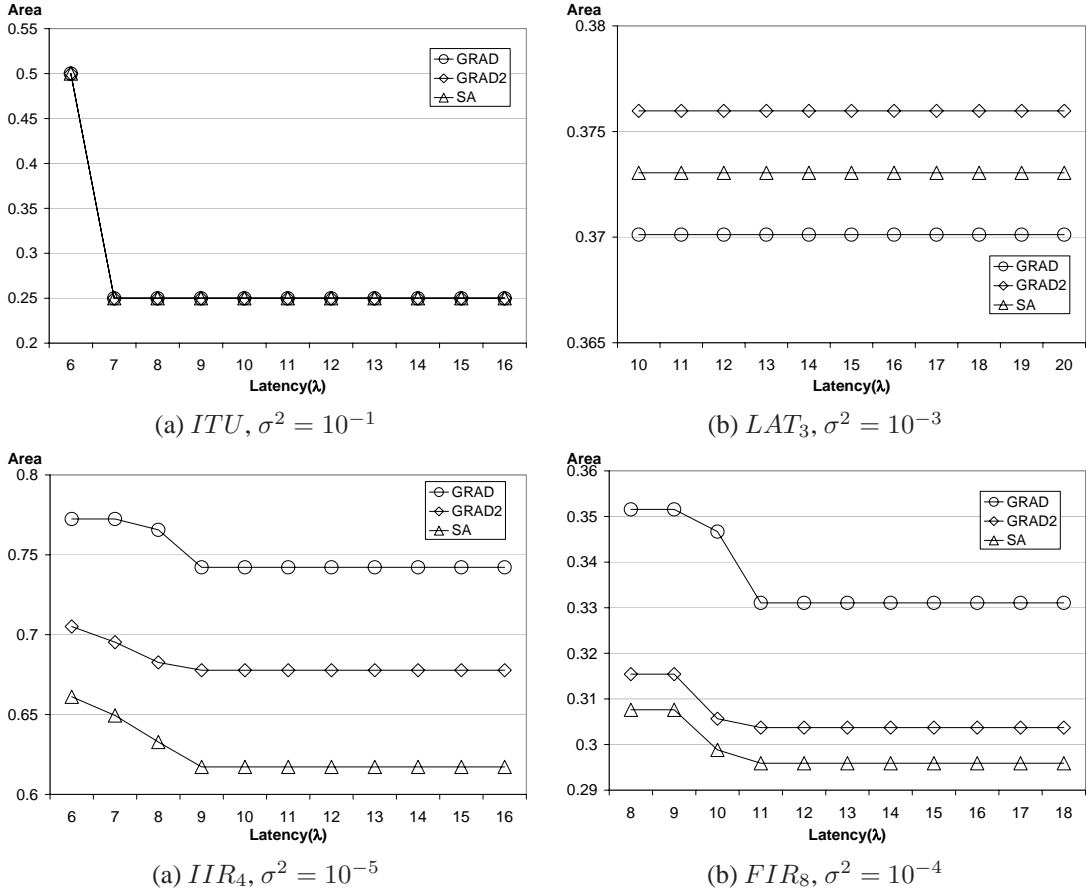


Figure 4.5: Resource-sharing HLS: Heterogeneous architectures

average area improvement of 2.72%, and SA 5.18%. However, the number of times that $GRAD$ exceeds $GRAD_2$ is greater than in the non-sharing case, and it reaches degradations of 5% sometimes. Moreover, SA , which always performed better in direct implementations, now manages to perform worse than $GRAD$, although to a negligible level (less than 1%). In general the mismatch between the expected behavior of algorithm is of 55.3%. Summarizing, complex WLA techniques applied before HLS, although performing better statistically, can have important performance degradation for very particular cases.

Fig. 4.5 depicts some area (∞ -norm) vs. latency results on the comparison of SA , $GRAD_2$ and $GRAD$ when resource sharing is allowed for heterogeneous-resource implementations. The latency ranges from $\lambda_{min}^{MWL-HET}$ to $\lambda_{min}^{MWL-HET} + 10$. Fig. 4.4-a and

Table 4.7: *GRAD* vs. *GRAD*₂, *SA*: resource-sharing heterogeneous architectures.

Bench.	σ^2	$\ \hat{\mathbf{A}}\ _\infty$ % (mean)		A_{LUT} % (mean)		A_{EMB} % (mean)	
		<i>GRAD</i> ₂	<i>SA</i>	<i>GRAD</i> ₂	<i>SA</i>	<i>GRAD</i> ₂	<i>SA</i>
<i>ITU</i>	10^{-1}	0.00	0.00	-3.62	-2.11	0.00	0.00
	10^{-2}	0.00	1.28	0.00	1.40	0.00	0.00
	10^{-3}	0.00	-4.91	0.00	-5.59	0.00	0.00
<i>LAT</i> ₃	10^{-3}	-1.58	-0.79	-1.58	-0.79	0.00	0.00
	10^{-4}	8.72	13.55	8.72	13.55	0.00	0.00
	10^{-5}	0.71	1.26	0.71	1.26	0.00	0.00
<i>IIR</i> ₄	10^{-3}	-1.34	-1.81	-2.12	-2.67	0.00	0.00
	10^{-4}	-4.01	-4.17	-4.01	-4.17	0.00	0.00
	10^{-5}	9.00	16.58	9.00	16.58	0.00	0.00
<i>FIR</i> ₈	10^{-3}	20.16	20.02	26.91	22.26	0.00	0.00
	10^{-4}	8.95	11.25	8.95	11.25	0.00	0.00
	10^{-5}	3.62	4.72	3.62	4.72	0.00	0.00
<i>All</i>		3.69	4.75	3.88	4.64	0.00	0.00

Fig. 4.4-b present a mismatch with respect to the expected behavior of direct implementations. For *IIR*₄ all word-length selection techniques perform the same in terms of ∞ -norm, since the embedded resources are masking the word-lengths differences between implementations. For *LAT*₃, *GRAD* is performing the best, despite the fact that it was *SA* which performed the best for direct implementations. Again, the area differences in these examples are negligible. Figures 4.5-c,-d show the expected behavior where *SA* performs better than the rest of algorithms, and *GRAD*₂ better than *GRAD*.

Table 4.6 contain implementation results of all benchmarks considering nine different quantization scenarios for heterogeneous implementations. The latencies range from $\lambda_{min}^{MWL-HET}$ to $\lambda_{min}^{MWL-HET} + 10$. The first column in the table contains the name of the benchmark. The second, the output noise variance applied. The third column holds the average ∞ -norm improvements obtained by *GRAD*₂ and by *SA* with respect to *GRAD*. The next two columns show similar values for the area of LUT-based and embedded resources.

Table 4.8: Optimization times for *GRAD*, *GRAD*₂ and *SA*.

Bench.	σ^2	<i>GRAD</i> (secs)	<i>GRAD</i> ₂ (secs)	<i>SA</i> (secs)
<i>ITU</i>	10^{-1}	0.078	0.20	26.47
<i>LAT</i> ₃	10^{-3}	1.30	4.01	38.03
<i>IIR</i> ₄	10^{-3}	1.89	6.02	33.08
<i>FIR</i> ₈	10^{-3}	1.03	1.62	65.27

The last row contains the total mean improvements.

*GRAD*₂ and *SA* improve the ∞ -norm area results 3.69% and 4.75% with respect to *GRAD*. There are 50% of mismatches between the expected WLA behavior for homogeneous direct implementations and resource-sharing heterogeneous implementation. And again there are cases where *GRAD*₂ and *SA* perform worse than *GRAD* and this was not expected. However, this time the degradation of *SA* is not negligible for some scenarios (*IIR*₄, $\sigma^2 = 10^{-4}$), reaching a maximum mean value of 4.17%. Also, it seems that the differences between word-length selection techniques is mainly due to variations to the LUT-based resources. LUT-based resources allow fine-grain architectural variations, while embedded resources only allow coarse-grain variations. The differences in terms of word-lengths of *GRAD*, *GRAD*₂ and *SA* are small, and therefore, embedded resources are not modified.

Optimization times

Table 4.8 contains information regarding typical WLA optimization times for *GRAD*, *GRAD*₂ and *SA*. These optimization times can be used as a measurement of the optimization techniques complexity. As expected *GRAD* is the less complex, followed by *GRAD*₂ and *SA*. Also, it can be seen that there is a clear correlation between the number of the signals present in the benchmarks (see table 3.2) and the time required to perform the optimization.

Summarizing, the traditional approach of performing WLA prior to HLS provides good

levels of optimization. However, there is a lack of control on the final results, and, in many cases the benefits of using complex and time-consuming word-length selection techniques, such as *SA*, are comparable or even worse than less complex and faster techniques such as gradient-descent based techniques. The optimality obtained when using direct architectures is no longer guaranteed when HLS is applied after WLA. The average improvements obtained by deep optimization techniques (i.e. *GRAD₂*, *SA*) are up to 5% for both homogenous and heterogenous implementations, which indicates that it might not be worth it to apply a very complex WLA (i.e. *GRAD₂*, *SA*), since the final architecture exploration does not fully explore the word-length reductions achieved.

4.5. Conclusions

WLA has been tackled in this chapter. A novel noise estimation method has been presented as well as a novel study of the impact of WLA on HLS.

The new noise estimation method has the following properties:

- Both LTI and non-linear algorithms are addressed.
- An optimized version able to handle LTI systems is available.
- Quantized forks' signals for both LTI and non-linear systems are handled.
- Very high accuracy:
 - Error within 1.85% for LTI and non-recursive non-linear algorithms for $\text{SQNR} \geq 40$ dB (mean error $< 0.94\%$).
 - Average error smaller than 3.64% for non-linear algorithms for $\text{SNQR} \geq 40$ dB.
- Extremely fast computational time (speedup ranging from x27 to x1210).

The results regarding the effect of the word-length selection techniques on HLS shows that:

- An MWL approach applied to non-sharing HLS obtains improvements of 70% over UWL for both homogeneous and heterogeneous architectures.
- The performance of a word-length selection technique obtained for homogeneous, direct implementation is not entirely kept after resource-sharing HLS:
 - The performance after HLS disagree for the 55.3% of experiments carried out for homogeneous implementations
 - The performance after HLS disagree for the 50% of experiments carried out for heterogeneous implementations
- *SA* word-length selection provides the best average area improvements.

Two main conclusions can be drawn: (i) the use of MWL word-length selection techniques during WLA highly reduces cost in comparison with UWL, and the application of fast noise estimation techniques is essential to this matter; and (ii) dividing the design process into two separate tasks (e.g. WLA and HLS) produces good results in comparison with an UWL approach, however it is not clear that complex MWL WLA provides much better results than simple MWL WLA after HLS.

CHAPTER 5

COMBINED APPROACH

This chapter addresses the problem of the combined application of WLA and HLS. First, an optimal approach is developed by means of an MILP description of the problem. The optimal results are used to extract conclusions about the combined problem particularities and they can be used as a starting point to assess the validity of novel combined WLA and HLS heuristic approaches. Second, an SA-based optimization procedure able to combine WLA and HLS is proposed. The optimizer is generated by integrating and extending the main two optimization procedures presented in chapters 3 and 4, as well as the conclusions extracted from the optimal analysis of the problem. It will be seen how the combined problem introduces new elements that require a thorough analysis of the new optimization procedure in order to achieve quasi-optimal results as well as to reduce computational complexity.

The chapter is divided as follows:

Section 5.1 presents an optimal approach to the problem. The section exposes the MILP description of the combined problem, as well as results and conclusions. Section 5.2 presents the SA-based combined WLA and HLS optimization procedure. Some results and conclu-

sions are also presented. Finally, in section 5.3 the conclusions are drawn.

5.1. Optimal approach

The optimal solution is sought by means of an MILP approach. The combined problem introduces too many variables in an MILP approach making its computation time extremely long. Some simplifications are introduced:

- Only LTI systems are addressed since the models of adders and gains are easily implemented using MILP.
- Only FUs are considered in the data path.
- Only 1-cycle latency resources are regarded.
- Only multipliers are shared.
- The area models of multipliers are approximated to ease their MILP representation.

The results from this section throw that, despite the simplifications, the computation times are still quite large for simple examples. The main point of the optimal analysis is twofold: (i) to provide some initial benchmarks to test the quality of HLS heuristic algorithms; and (ii) to provide insights on the combined application of WLA and HLS that might help in designing efficient HLS heuristic algorithms.

The combined MILP description of the problem is partly based on the MILP-based optimal WLA presented in [CCL03a] and on the MILP-based optimal HLS in [CCL00b] and it has been published in [CCC⁺06]. The combined application of these two design techniques involves the introduction of a dynamic compatibility graph that is not present in any of the techniques by themselves. Moreover, since word-lengths are also part of the optimization problem, the area of resources changes dynamically as well.

5.1.1. Problem definition

The combined application of the word-length allocation and architectural synthesis tasks has as a starting point a computation graph $G_{SFG}(V, S)$, a maximum latency λ , and a maximum noise variance E_{max} at the output.

We consider the set of operations $V = V_G \cup V_A \cup V_D \cup V_F \cup V_I \cup V_O$ composed of gains, additions, unit delays, forks (branching nodes) and input and output nodes. Signals are in two's complement fixed-point format defined by the pair (n, p) , where n is the word-length of the signal not including the sign bit, and p is the scaling of the signal that represents the displacement of the binary point from the sign bit (see Fig. 3.2). The scaling information of signals is assumed to be available, so the value of p is known.

Operations are to be implemented on resources from set R and it is the aim of the combined approach to find the word-lengths n , the time step when each operation is executed (*scheduling*), the types and number of resources forming R (*resource allocation*) and the binding between operations and resources (*resource binding*) that complies with both λ and E_{max} constraints, while achieving minimum area.

We adopt the quantization error presented in [CCL99]. The quantization error introduced by the quantization of a signal s_i from n_i^{pre} bits to n_i bits ($n_i^{pre} \leq n_i$) is modeled by the injection of a uniform-distributed white noise with a variance equal to eqn. 4.1. The variance of the noise contribution at the output was expressed in eqn. 4.3 and can be rewritten in terms of the L_2 -norm ($L_2(\cdot)$):

$$\sigma_o^2 = \sum_{i=0}^{|S|-1} \sigma_i^2 \cdot \frac{1}{2\pi} \int_{-\pi}^{\pi} |G_i(e^{j\Omega})| d\Omega = \sum_{i=0}^{|S|-1} \sigma_i^2 \cdot L_2(G_i(Z)) \quad (5.1)$$

As stated in [CCL03a] and mentioned in chapter 4 (see 4.1, Fig. 4.3), the error introduced by forks requires a special treatment. The error that a w-way fork introduces can be expressed

as in eqn. 5.2, where the w -tuple expresses the order of the outputs [CCL03a].

$$\begin{aligned}
& \bigwedge_{r=1}^{w-1} n_{\alpha(r)} \geq n_{\alpha(r+1)} \Rightarrow \\
E_v = & \frac{2^{2p_i}}{12} \left(\sum_{r=1}^{w-1} L_2^2 \left(\sum_{h=r+1}^w G_{\alpha(h)} \right) \right) (2^{-2n_{\alpha(r+1)}} - 2^{-2n_{\alpha(r)}}) \\
& + L_2^2 \left(\sum_{h=1}^w G_{\alpha(h)} \right) (2^{-2n_{\alpha(1)}} - 2^{-2n_i^{pre}}) \quad (5.2)
\end{aligned}$$

The data-flow of a single iteration of the algorithm is expressed by means of the sequencing graph $G_{SEQ}(O, E)$ extracted from G_{SFG} (see 3.1).

In our approach we assume 1-cycle latency operations. Each operation $v \in V$ with an assigned node $o = \theta(v)$ can be executed during the time interval defined by $T(o)$ (eqn. 5.3) where Z_+ denotes the set of non-negative integers. $ASAP(o)$ is the execution time of operation o for the *as soon as possible* scheduling, and $ALAP(o, \lambda)$ is the execution time of operation o for the *as late as possible* scheduling for a total time steps of λ . The set of all possible execution times is given by eqn. 5.4. Input data is supposed to be available, as well as delays' data, so nodes $o \in O_I \cup O_{DR}$ are scheduled at time $T(o) = -1$. Also, output data is available at time step λ . Delay writings, are also scheduled at time step λ , to simplify the analysis.

$$T(o) = \{t \mid t \in Z_+ : t \geq ASAP(o) \wedge t \leq ALAP(o, \lambda)\} \quad (5.3)$$

$$T = \{o \mid \exists o \in O \setminus O_I \cup O_O \cup O_{DR} \cup O_{DW} : t \in T(o)\} = \{1 \dots \lambda\} \quad (5.4)$$

The set of resources $R = R_M \cup R_A \cup R_{REG}$ is divided into multipliers, adders and registers which implement gains, additions and delays. Multiplexing logic and memory to

store intermediate values are not considered among resources. We express the compatibility between an operation, or set of operations, and resources with function $R(v) : V \rightarrow R$.

Since only multipliers are regarded as shareable, there are dedicated resources to implement each addition and delay. Thus, $R_A(V_A)$ and $R_R(V_D)$ are one-to-one functions and $|R_A| = |V_A|$ and $|R_R| = |V_D|$.

Multipliers have one input devoted to the gain coefficients and its word-length is equal to a system-wide constant, n_{coeff} . The other input is assigned to the input signal of gains and must have a word-length greater than or equal to the maximum word-length of the input signals of gains bound to the resource.

An upper bound on the number of necessary multipliers can be estimated from the number of multipliers required to implement the *ASAP* scheduling. Initially, each gain is implemented in one multiplier, therefore $R_M(V_G) = R_M$.

The cost of an adder $r \in R_A$ bound to addition $v \in V_A$ with inputs s_i and s_j and output s_o is kept as in eqns. 3.14-3.16.

The cost in slices of a register $r \in R_R$ bound to delay $v \in V_D$ with input s_i is given by the straight-forward equation 5.5.

$$r \in R_R, o \in V_D : R(v) = r, \text{cost}(r) = 0.25 \cdot n_i \quad (5.5)$$

Equation 5.6 contains the cost of a multiplier $r \in R_M$ bound to a subset of gains $V'_G \subseteq V_G$ with inputs $S' = in(V'_G) \subseteq S$.

$$\text{cost}(r) = 0.5 \cdot (\max(n(S')) + 1) (n_{coeff} + 1) \quad (5.6)$$

5.1.2. MILP formulation

This section relies on some knowledge of integer linear programming [GN72]. The variables used in the MILP model are divided into: binary scheduling and resource binding variables (x), integer signal word-lengths (n), integer signal word-lengths before quantization (n^{pre}), binary auxiliary signal word-lengths (\bar{n}), binary auxiliary signal word-lengths before quantization (\bar{n}^{pre}), binary decision variables (δ , ϵ and η), integer adder costs (A), integer auxiliary variables (β) and real fork node error variables (E).

The objective function is the sum of the area of all resources (adders, registers and multipliers) and it is given by 5.7. The cost of adders A_r is to be linearized in the constraints section according to eqn. 3.14.

$$\min : \sum_{r \in R_A} A_r + 1/4 \sum_{r \in R_R} n_{in(r)} + 1/2 \sum_{r \in R_M} (n_r + 1) (n_{coeff} + 1) \quad (5.7)$$

The constraints related to scheduling, resource allocation and resource binding are steered by means of the binary variables $x_{o,t,r}$ shown in eqn. 5.8.

$$x_{o,t,r} = \begin{cases} 1, & \text{if operation } o \text{ is scheduled at time-step } t \text{ on resource } r \\ 0, & \text{otherwise} \end{cases} \quad (5.8)$$

Equation 5.9 shows the binding constraint that ensures that an operation is executed on exactly one resource. The next constraint 5.10 states that a resource does not implement more than one operation at a time. Note that there is no need to apply 5.9 and 5.10 to operations with dedicated resources. The precedence constraints are given by 5.11 ensuring

that operations obey the dependencies in the sequencing graph G_{SEQ} .

$$\forall v \in V_G, \sum_{r \in R(v)} \sum_{t \in T(o=\theta(v))} x_{o,t,r} = 1 \quad (5.9)$$

$$\forall t \in T, \forall r \in R_M, \sum_{v \in V: r \in R(v)} \sum_{t_1 \in \{t\} \cap T(o=\theta(v))} x_{o,t_1,r} \leq 1 \quad (5.10)$$

$$\begin{aligned} & \forall (o_1, o_2) \in E, \forall t \in T(o_2) \cap T(o_1) \\ & \sum_{r \in R(\theta^{-1}(o_2))} \sum_{t_2 \in T(o_2): t_2 \leq t} x_{o_2,t_2,r} + \sum_{r \in R(\theta^{-1}(o_1))} \sum_{t_1 \in T(o_1): t_1 \geq t} x_{o_1,t_1,r} \leq 1 \end{aligned} \quad (5.11)$$

And finally, eqn. 5.12 expresses the resource compatibility constraints, which guarantee that a resource bound to several operations must be compatible with all of them. Again, only multipliers are considered: the input devoted to signals must have a word-length as big as the maximum of the word-lengths of each gain input bound to it. The summation $\sum x_{o,t,r}$ is equal to 1 if operation $v = \theta(o)$ is bound to resource r .

$$\forall r \in R_M, \forall v \in V_G, n_{in(v)} - n_r \leq \hat{n}_{in(v)} \left(1 - \sum_{t \in T(o=\theta(v))} x_{o,t,r} \right) \quad (5.12)$$

Note that although only multipliers are prone to sharing the notation can be easily extended to include more resources that can be shared (dividers, adders, etc.) or to map more than one type of operation to the same resource (e.g. gains and multiplications bound to multipliers).

The linearization of the cost of an adder $v \in V_A$ with inputs s_i and s_j and output s_o is based on the model from [CCL03a]. Constraints 5.13-5.16 cast 3.15, 3.16 using binary decision variables δ_{a_1} and δ_{a_2} , and also trivial bounds on the left side of the equations. Equation 3.14 is directly implemented using constraint 5.17.

$$n_i - n_j + p_j - p_i < \delta_{a_1}(\hat{n}_i + p_j - p_i) \quad (5.13)$$

$$\beta_v - n_j + p_j - p_i + 1 \geq (1 - \delta_{a_1})(-\hat{n}_j + p_j - p_i + 1) \quad (5.14)$$

$$n_i - n_j + p_j - p_i \geq \delta_{a_2}(-\hat{n}_j + p_j - p_i) \quad (5.15)$$

$$\beta_v - n_j + 1 \geq (1 - \delta_{a_2})(-\hat{n}_j + 1) \quad (5.16)$$

$$A_r = (\beta_v + 1 - m_v) / 2 \quad (5.17)$$

Now, the constraints related to the estimation of the noise at the output of the system [CCL03a] are presented. The error constraint is given by 5.18 and it is divided into two summations, the first dealing with signals from forks, and the second dealing with the remaining signals in S (see section 4.1). The preceding operations of a given operation v are

$$\sum_{v \in V_F} E_v + \sum_{s_i \in S \setminus \text{in}(V_F) \setminus \text{out}(V_F)} \frac{2^{2p_i}}{12} L_2^2(G_i(Z)) \left(2^{-2n_i} - 2^{-2n_i^{pre}} \right) \leq E_{max} \quad (5.18)$$

Note that non-constant powers of two must be linearized. Each term 2^{-2n} is replaced by $\sum_{b=1}^{\hat{n}} 2^{-2b} \cdot \bar{n}_b$, where \bar{n}_b are binary auxiliary variables associated to signal n by 5.19 and 5.20. For simplification' sake, all non-constant powers of two are left as they are throughout the text.

$$n - \sum_{b=1}^{\hat{n}} b \cdot \bar{n}_b = 0 \quad (5.19)$$

$$\sum_{b=1}^{\hat{n}} \bar{n}_b = 1 \quad (5.20)$$

The noise E_o introduced by a fork v is expressed by constraints 5.21-5.24, which come from applying DeMorgan's theorem to 5.2 and linearizing the disjunction obtained. Binary

variables η and ϵ are introduced. These constraints are repeated for each possible ordering α of the outputs of a fork.

$$n_{\alpha(1)} - n_{\alpha(2)} < \epsilon_{v,\alpha(1),\alpha(2)} \hat{n}_{\alpha(1)} \quad (5.21)$$

...

$$n_{\alpha(w-1)} - n_{\alpha(w)} < \epsilon_{v,\alpha(w-1),\alpha(w)} \hat{n}_{\alpha(w-1)} \quad (5.22)$$

$$\begin{aligned} E_v - \frac{2^{2p_i}}{12} \sum_{r=1}^{w-1} L_2^2 \left(\sum_{h=r+1}^w G_{\alpha(h)} \right) (2^{-2n_{\alpha(r+1)}} - 2^{-2n_{\alpha(r)}}) \\ - \frac{2^{2p_i}}{12} L_2^2 \left(\sum_{h=1}^w G_{\alpha(h)} \right) (2^{-2n_{\alpha(1)}} - 2^{-2n_i^{pre}}) \geq \\ -\eta_{v,\alpha} \frac{2^{2(p_i-1)}}{12} \sum_{r=0}^{w-1} L_2^2 \left(\sum_{h=r+1}^w G_{\alpha(h)} \right) \end{aligned} \quad (5.23)$$

$$\sum_{r=1}^{w-1} \epsilon_{v,\alpha(r+1)} + \eta_{v,\alpha} \leq w - 1 \quad (5.24)$$

The last set of constraints computes the word-lengths before quantization when considering scaling and word-length propagation information [CCL03a].

Given an addition o with inputs s_i and s_j and output s_o , its output's word-length is equal to $\max(n_i - p_i + p_o, n_j - p_j + p_o)$, expression linearized through 5.25 and 5.26.

$$n_o^{pre} \geq n_i - p_i + p_o \quad (5.25)$$

$$n_o^{pre} \geq n_j - p_j + p_o \quad (5.26)$$

Delays o with input s_i are conditioned through 5.27.

$$n_o^{pre} = n_i \quad (5.27)$$

Regarding forks, the outputs do not require conditioning but its inputs must comply with

$$\forall s_i \in \text{out}(v) n_{input} \geq n_i. \quad (5.28)$$

The conditioning of gain v with input s_i and output s_o is expressed by constraint 5.29, where $p_{coeff}(v)$ is the scaling of the coefficient associated to o .

$$n_o^{pre} = n_i + n_{coeff} - p_i - p_{coeff}(v) + p_o \quad (5.29)$$

Finally, 5.30 indicates that signals must be truncated to a word-length smaller than or equal to its pre-quantization word-length.

$$\forall s_i \in S : \exists (n_i, n_i^{pre}), n_i \leq n_i^{pre} \quad (5.30)$$

Bounds on word-lengths \bar{n}_s are estimated using an adaptation of the procedure presented in [CCL03a]:

1. Use a heuristic algorithm (i.e. gradient-descent WLA) to allocate word-lengths and calculate the area A due to gains.
2. Assign to each gain input the word-length that makes its area to be as big as A .
3. Set all gain inputs to the maximum word-length of all gain inputs.
4. Condition the graph.

The rationale is that an over-sized quantization of the algorithm can be used to set the upper bounds of word-lengths, since it is highly probable that word-lengths tend to decrease throughout WLA.

Table 5.1: Area reduction (%) obtained by the optimal combined approach

E_{max} (σ^2)	FIR_2 8x4 $\lambda = 4$	FIR_3 8x3 $\lambda = 4$	FIR_4 4x4 $\lambda = 4$	IIR 4x4 $\lambda = 4$	IIR 4x4 $\lambda = 6$
1.0e-5	0.00 (31)*	6.72 (59.5/55.5)**	0.00 (27.5)	-	-
1.1e-5	4.84 (31.5/29.5)	5.93 (59/55.5)	0.00 (29)	-	-
3.3e-5	0.00 (27.5)	1.90 (52.5/51.5)	0.00 (28.5)	-	-
5.0e-5	0.00 (27)	7.62 (52.5/48.5)	0.00 (28.5)	-	-
1.0e-4	0.00 (24)	1.08 (46/45.5)	0.00 (28)	-	-
1.1e-4	0.00 (28.5)	1.08^C (46/45.5)	0.00 (28)	-	-
3.3e-4	0.00 (20)	2.50 (40/39)	5.56 (27/25.5)	-	-
5.0e-4	0.00 (19.5)	0.00 (38.5)	0.00 (25)	0.00 (40)	-
1.0e-3	10.53 (19/17)	11.84 (38/37.5)	0.00 (24)	7.69 (39/36)	-
1.1e-3	13.16 (19/16.5)	5.80^B (34.5/32.5)	7.84 (25.5/23.5)	9.09 (38.5/35)	3.70 (27/26)
3.3e-3	0.00 (15)	12.70 (31.5/27.5)	4.65 (21.5/20.5)	0.00 (32)	0.00 (22)
5.0e-3	10.71 (14/12.5)	0.00 (25.5)	0.00 (18.5)	0.00 (28)	0.00 (20)
1.0e-2	0.00 (11.5)	12.00 (25/22)	2.78 (18/17.5)	0.00 (26)	0.00 (18)
1.1e-2	0.00 (11.5)	12.24^A (24.5/21.5)	2.78 (18/17.5)	0.00 (26)	0.00 (18)
3.3e-2	0.00 (8.5)	5.40 (18.5/17.5)	0.00 (12.5)	13.04 (23/20)	6.67 (15/14)
5.0e-2	0.00 (8)	0.00 (14.5)	0.00 (12)	0.00 (19)	0.00 (11.5)
1.0e-1	0.00 (8)	0.00 (14)	0.00 (11.5)	11.43 (17.5/15.5)	0.00 (11.5)
* 0% improvement (Area)			** % Improvement (A_{SEQ}/A_{COMB})		

5.1.3. Results and conclusions

This subsection presents the comparison results between the sequential application of WLA and HLS and its combined application using optimal techniques. It is sought to quantize the benefits of the combined approach in terms of area reductions, and also to extract possible heuristic rules that can be applied to the synthesis of DSP circuits.

An MILP solver [MOS] was used to find the optimal solutions for a set of FIR and IIR filters. The filters coefficients were obtained using the tool *fdatool* from Matlab 6.5 [Mat]. The FIR filters were implemented using the direct transposed symmetric FIR structure. We denote FIR_2 a second order FIR filter with 8-bit inputs and 4-bit coefficients

Table 5.2: Detailed word-lengths for FIR_3 .

%	g_{1-3}	g_2	m_1	r_1	r_2	r_3	a_1	a_2	a_3
12.24 ^A	$2 \times 8 \rightarrow 3 \times 8$	$4 \times 8 \rightarrow 3 \times 8$	$4 \times 8 \rightarrow 3 \times 8$	3	3	3	5	$4 \rightarrow 5$	$5 \rightarrow 6$
5.80 ^B	$3 \times 8 \rightarrow 5 \times 8$	5×8	5×8	$5 \rightarrow 4$	$5 \rightarrow 4$	$5 \rightarrow 4$	$7 \rightarrow 6$	5	8
1.08 ^C	$5 \times 8 \rightarrow 7 \times 8$	7×8	$7 \rightarrow 8$	6	6	6	8	$7 \rightarrow 6$	9

$B_{FIR_2} = [0.1759, 0.8, 0.1759]$; FIR_3 a 3rd order FIR filter with 8-bit inputs and 8-bit coefficients $B_{FIR_3} = [0.1172, 0.6013, 0.6013, 0.1172]$; and FIR_4 a 4th order FIR filter with 4-bit inputs and 4-bit coefficients $B_{FIR_4} = [0.1210, 0.1423, 0.85, 0.1423, 0.1210]$. The IIR filter (IIR) was a 2nd order filter with 4-bit inputs, gain $G = 0.307089$ and 4-bit coefficients $SOS = [1.0, 1.9999, 0.9999, 1.0, 0.0640955, 0.314]$, implemented using the direct form II transposed. The filters were solved for different latencies and for each latency two solutions were computed, one for the sequential approach, where the error constrained problem was solved first and its solution was fed to the latency constrained problem, and another for the combined approach.

The comparison results are in table 5.1 in terms of percentage of area reduction comparing both sequential and combined approaches. The number of slices required for the different approaches is also provided (sequential/combined). The area savings range from 0% to 13.16%, and are due to an optimal exploration of the dependencies between word-lengths, resources and error variance. Empty cells imply that a solution was not found by the MILP solver in practical times (less than 12 hours). It can be seen, how the improvements are obtained sporadically. All examples except FIR_3 present sporadic improvements, and it seems to be no relationship between the error constraints and the occurrence of an area improvement. The advantage of the combined approach appears when a particular combination of latency, error constraint, and the very data dependencies of the DSP algorithm allows the trade-off between area and quantization noise, and due to the many variables involved, it looks extremely difficult to predict when this situation may arise.

Table 5.2 shows the word-lengths, including the sign bit, assigned to gains ($g_{1,3}$ and g_2) and to multipliers (m_1), adders (a_1 , a_2 and a_3) and registers (r_1 , r_2 and r_3) for the error/latency conditions A , B and C (see 5.1, third column) for FIR_3 . The first row represents the area saving and states the error/latency condition (A , B and C). The rest of rows show the word-lengths, showing two word-lengths if the sequential results differ from the combined results (sequential/combined). In case A the area of the multiplier m_1 is reduced while the area of adders is slightly increased. Thus, it is possible to compensate the noise introduced by a resource size reduction by another resource size increase. Note that g_{1-3} is increased while g_2 is decreased, and the overall effect is that the area of the multiplier is reduced. It looks interesting to perform a simultaneous decrease and increase of the word-lengths of a resource's signals to minimize the effect of the quantization noise (since a word-length decrease tends to add noise, while a word-length increase tends to reduce noise). In case B the area of registers and adders is reduced thanks to the increase of the word-lengths of gains. Note, that the increase in the size of gains does not produce a reduction in the multiplier's size, but it reduces quantization noise, allowing to add some noise by means of decreasing another resources size. Finally, case C shows that an increase in the word-length of gains enables reducing the area of adders. Again, the key here is an increase in the signals related to m_1 that reduces noise, but does not increase area.

Heuristic rules

Some possible optimization rules for a heuristic that combines WLA and HLS can be proposed from the previous results:

- A shared resource with operations with sizes smaller than the resource size can be used to reduce noise by means of increasing all operations to the maximum permitted size. An increase in word-length (size) produces a decrease in noise, which allows a possible resource size reduction, and therefore, a noise increase, in further optimization steps.

- A shared resource with operations with sizes smaller than the resource size can be used to reduce the resource area by means of setting the size of operations to the maximum permitted size minus one unit.
- A combination of the two previous steps.
- Simultaneous increase and decrease of the size of several resources.

The execution times to solve the MILP problems range from several seconds (FIR_2) to several hours (IIR) using a PC with a 3.2 GHz Pentium-IV and 2 GB of RAM.

Summarizing, area improvements up to a 13% are shown when applying a combined WLA and HLS approach. The optimal analysis performed shows that, for the particular conditions of the analysis, the area improvements provided by the combined approach occur sporadically. Some clues about the possible modifications to the word-lengths of the algorithm and the data-path architecture that some heuristic optimization procedures may apply, are provided.

5.2. Heuristic approach

The proposed heuristic approach is an extension of both SA-based HLS and WLA optimization methods presented in chapters 3 and 4, respectively. Again, the combination of these two techniques is not as simple as the combination of the SA movements. For instance, HLS is performed based on the knowledge of the area and latency of resources, as well as on the compatibility between operations and resources. A permanent change in the word-lengths requires a continuous updating of the set of available resources R as well as the compatibility graph. This adds a new dimension to the problem. A technique to minimize the

Algorithm 5.1 Combined optimization procedure

Input: Unquantized $G_{SFG}(V, S)$, λ , E_{max} **Output:** Quantized $G_{SFG}(V, S)$, $R = R_{FU} \cup R_{REG} \cup R_{MUX}$, φ , β , γ

- 1: Extract $G_{SEQ}(O, E)$
 - 2: Find initial set of n , $N_0 = \{n_0, \dots, n_{|S|-1}\}$, by means of *UWL WLA* (4.3)
 - 3: Extract $G_{COMP}(O, R_{FU})$, R_{FU} , R_{REG}
 - 4: Find initial mapping m_0
 - 5: Compute initial area \mathbf{A}_0 from G_{SEQ} and m_0
 - 6: $\mathbf{A}_{min} = \mathbf{A}_{last} = \mathbf{A}_0$
 $m_{min} = m_{last} = m_0$
 $N_{min} = N_{last} = N_0$
 $T = T_0$
 $iteration = accepted = exit = 0$
 - 7: **while** $\neg exit$ condition **do**
 - 8: $m = m_{last}$, $N = N_{last}$
 - 9: $iteration = iteration + 1$
 - 10: Perform change to current m and/or N and compute area \mathbf{A} (see Alg. 5.2)
 - 11: $\Delta \mathbf{A} = \frac{\|\hat{\mathbf{A}}_{last}\|_n - \|\hat{\mathbf{A}}\|_n}{\|\hat{\mathbf{A}}_0\|_n}$
 - 12: Perform standard SA (see Alg. 3.1, lines 10- 29)
 - 13: **end while**
-

computational complexity of the combined optimization procedure is suggested for the sake of reducing convergence time.

5.2.1. Optimization procedure

The optimization procedure follows the same scheme depicted in Alg. 3.1: a loop where changes to the cost function are performed and are accepted with some probability (cost reductions are always accepted). However, there are some modifications. Alg. 5.1 shows the main modifications introduced to the SA-based optimization procedure. First, the movements are applied now to both mapping m and the set of word-lengths $N = \{n_0, \dots, n_{|S|-1}\}$. Therefore, the initial solution is composed of the couple (N_0, m_0) , where N_0 is the result of applying an UWL quantization (alg. 4.3) to a LUT-based direct implementation of the algorithm, and m_0 is a fastest-resource mapping (see lines 2-5). Thus, a deeper design space

Algorithm 5.2 Movements to cost function

Input: $G_{SFG}(V, S), G_{COMP}(O, R_{FU}), m, iterations$ **Output:** $R_{FU}, \varphi, \beta, \gamma, A$

```
1: if  $iterations \% M$  then
2:    $m_{type} = \neg m_{type}$ 
3:   if  $m_{type} = M_{HLS}$  then
4:     Update  $G_{COMP}$ 
5:   end if
6: end if
7: if  $m_{type} = M_{HLS}$  then
8:   Perform HLS movement
9:   Compute area  $A$  from  $m$  (Alg. 5.3)
10: else
11:   Perform WLA movement
12:   if  $m$  is kept then
13:     Compute area  $A$  (call Alg. 5.3 from line 2)
14:   else
15:     Compute area  $A$  from  $m$  (Alg. 5.3)
16:   end if
17: end if
```

exploration is now permitted, since the initial set of word-lengths is *far* from a local minimum. After the initial solution is calculated and the variables are initialized (line 6), the optimization loop is entered. Note that the application of the movements as well as the computation of the area are now grouped into a single line (10). Then, the area is normalized and the procedure carries on as in standard SA optimization. Line 10 has been developed in algorithms 5.2 and 5.3. Let us first highlight that the computation of the area (Alg. 5.3) is very similar to that of Alg. 3.2 but it is necessary to include the penalty α_E to account for solutions where the noise constraints are not met (see lines 7- 15). Now, there is a double penalty to solutions that do not comply with the latency constraint (λ) and/or with the noise constraint (E_{max})

In order to explain Alg. 5.2, which generates the movements in the current solution and computes the new cost function, it is necessary to clarify some concepts before. In Alg. 3.1 the changes on m would require the computation of a new binding β and a new scheduling

Algorithm 5.3 Computation of area cost

Input: $G_{SFG}(V, S), G_{SEQ}(O, E), m, E_{max}, \lambda$ **Output:** $R = R_{FU} \cup R_{REG} \cup R_{MUX}, \varphi, \beta, \gamma, \mathbf{A}$

- 1: Call Alg. 3.2 up to line 26 (computation of A' and α_λ)
 - 2: Extract D, R_{REG}
 - 3: Compute register binding
 - 4: Extract R_{MUX}
 - 5: $R = R_{FU} \cup R_{REG} \cup R_{MUX}$
 - 6: $\mathbf{A}' = \sum_{r \in R} \mathbf{A}(r) \cdot inst(r)$
 - 7: $E' = \text{Noise at the outputs of } G_{SFG}$
 - 8: **if** $E' < E_{max}$ **then**
 - 9: $E' = E_{max}$
 - 10: **end if**
 - 11: $\alpha_E = (E' - E_{max}) / E_{max}$
 - 12: $\mathbf{A} = \mathbf{A}' \cdot (1 + \alpha_\lambda + \alpha_E)$
 - 13: **if** $\|\hat{\mathbf{A}}\|_n < \|\hat{\mathbf{A}}_{min}\|_n$ **then**
 - 14: $\mathbf{A} = \mathbf{A}_{min} \cdot (1 + \alpha_\lambda + \alpha_E)$
 - 15: **end if**
-

of operations (φ, γ) , while in the equivalent for WLA (section 4.3) the changes produced to N would require the conditioning of graph G_{SFG} and the computation of the new noise at the outputs of the graphs. However, in the combined problem a change in N may produce a change in the set of resources (i.e. a resource's size is reduced or increased), in m (i.e. two resources that had different sizes are now equivalent) and in G_{COMP} . Therefore, both the computation of the noise and the new values for β, φ and γ are required. The cost function (area) could be computed following Alg. 3.2, but R, G_{COMP} and m must be updated before it is called. The continuous updating of these elements introduces a computational overhead, not present in the previous SA-based approaches.

In order to reduce this overhead, algorithm 5.2 is proposed. The movements are divided into HLS movements (M_{HLS}) and WLA movements (M_{WLA}). Movements of the same type are applied during M consecutive iterations (lines 1-6). After that, the type of movement is toggled.

HLS movements do not change the word-lengths n , so they behave as stated in 3.3. The

HLS movements are:

- M_{HLS}^A : Map an operation $o \in O$ to a non-mapped resource.
- M_{HLS}^B : Map an operation o to another already mapped resource.
- M_{HLS}^C : Swap the mapping of two compatible operations mapped to different resources.

The WLA movements are:

- M_A^{WLA} : Select a signal $s_i \in S \setminus in(V_F) \cup out(V_D)$ and increase one unit n_i .
- M_{WLA}^B : Select a signal $s_i \in S \setminus in(V_F) \cup out(V_D)$ and decrease one unit n_i .
- M_{WLA}^C : Select a resource $r \in R_{FU}$ and increase one unit the *size* of all operations bound to it that do not increase the original size of the resource.
- M_{WLA}^D : Select two resources $r_1, r_2 \in R_{FU}$. Apply movement M_{WLA}^C to r_1 and r_2 . Reduce one unit the size of operations bound to r_2 that meet the original size of the resource.

The probability of occurrence of M_{HLS} and M_{WLA} is 0.5. Each movement from M_{HLS} is equiprobable (0.5/3), while the movements from M_{WLA} has the following probabilities: M_A^{WLA} 0.5/2, M_B^{WLA} 0.50/4, and M_C^{WLA} and M_A^{WLD} 0.4/8%. As in 4.3, any word-length decrease is favoured.

These movements may require updating R , m and G_{COMP} . However, updating G_{COMP} is only necessary when switching from M_{WLA} to M_{HLS} (see Alg. 5.2, line 4).

Movement M_{WLA}^A may not result in an increase in the size of resources. However, the area due to multiplexers and registers may change, so it is necessary to modify the current area by executing Alg. 5.3 from line 2. If the size of resources has been changed, all operations bound to these resources are mapped to the minimum area resource that can execute them. Therefore, a change in m is produced. If the resource is not present in R_{FU} , it is

added, so R_{FU} is also changed. These changes force a new computation of the area cost by means of Alg. 5.3. A similar situation arises for movement M_{WLA}^B .

Movement M_{WLA}^C does not change the size of resource r , but the area of multiplexers and registers may vary. Thus, it is necessary to execute Alg. 5.3 from line 2.

Finally, movement M_{WLA}^D always requires the execution of the whole Alg. 5.3, and it is also possible that new FUs are added to R_{FU} .

Movements M_{WLA}^C and M_{WLA}^D are genuine to the combined problem and are selected due to the final conclusions obtained from the optimal analysis in the previous section. M_{WLA}^C keeps the area of a resource, but increases the word-lengths of the signals directly related to it. Thus, there is a chance of reducing noise with no FU area penalty. In movement M_{WLA}^D , the noise introduced by a resource area reduction is somehow compensated by applying movement M_{WLA}^C to other resource, therefore, reducing noise. Note that the reduction of the FU area in M_{WLA}^D not only reduces word-lengths but it also increases them, trying to minimize the noise introduced.

Note that M_{WLA} movements only update the set of resources R and change the mapping m , which is a trivial task. However, updating G_{SFG} requires checking that the operations are still compatible to the resources that they were linked before word-length changed, and also checking if they are compatible to the new resources introduced. This task is now performed each $2 \cdot M$ optimization steps (see Alg. 5.2, line 4), with the consequent computational efficiency improvement.

5.2.2. Results and conclusions

This subsection presents the comparison results between the sequential application of WLA and HLS and its combined application using an SA-based technique. The results and conclusions of the optimal analysis were limited to small benchmarks as well as architectural constraints. Now, the experiments are based on a more realistic architectural model and more

Table 5.3: Characteristics of benchmarks.

Alg.	IN	OUT	+	*	Forks	Delays	Nodes	+/*	Signals
<i>ITU</i>	3	3	4	6	4	0	20	10	21
<i>LAT₃</i>	1	1	8	9	9	3	31	17	36
<i>IIR₄</i>	1	1	8	7	6	4	27	15	34
<i>FIR₈</i>	1	1	8	5	8	8	31	13	38
<i>DCT₄</i>	4	4	8	8	8	0	32	16	36
<i>DCT₈</i>	8	8	32	22	32	0	102	54	120

complex benchmarks. Again, it is sought to quantize the benefits of the combined approach aiming at providing the grounds to develop synthesis algorithms able to handle industrial designs.

The benchmarks used are *ITU*, *LAT₃*, *IIR₄*, *FIR₈*, with the addition of a 4-point DCT transform (*DCT₄* [Par99]), and an 8-point DCT transform (*DCT₈* [Par99]). All algorithms are assigned 8-bit inputs and 12-bit constant coefficients. Table 5.3 contains information about the number of operations and signals of the benchmarks. The algorithms have been tested under different latency and output noise scenarios assuming a system clock of 125 MHz, as in section 3.4. The target devices belong to the Xilinx Virtex-II family, being the area results normalized with respect to the XC2V40 device.

The MWL HLS results from chapter 3, which followed a sequential approach (*SEQ*), are compared to the combined WLA and HLS approach (*COMB*) presented in this section. In the sequential approach, the tasks of WLA and HLS are performed separately, while in the combined approach they are executed in an intertwined manner. All SA-based optimizations (WLA, HLS and combined WLA-HLS) are repeated four times and the best results are selected (as explained in 3.4).

Figures 5.1, 5.2 and 5.3 depict the results of the comparison of *SEQ* vs. *COMB* implementation for homogeneous-resource FPGAs. The subfigures are arranged in couples, which are related to the same benchmark. The left subfigures depict the area improvement

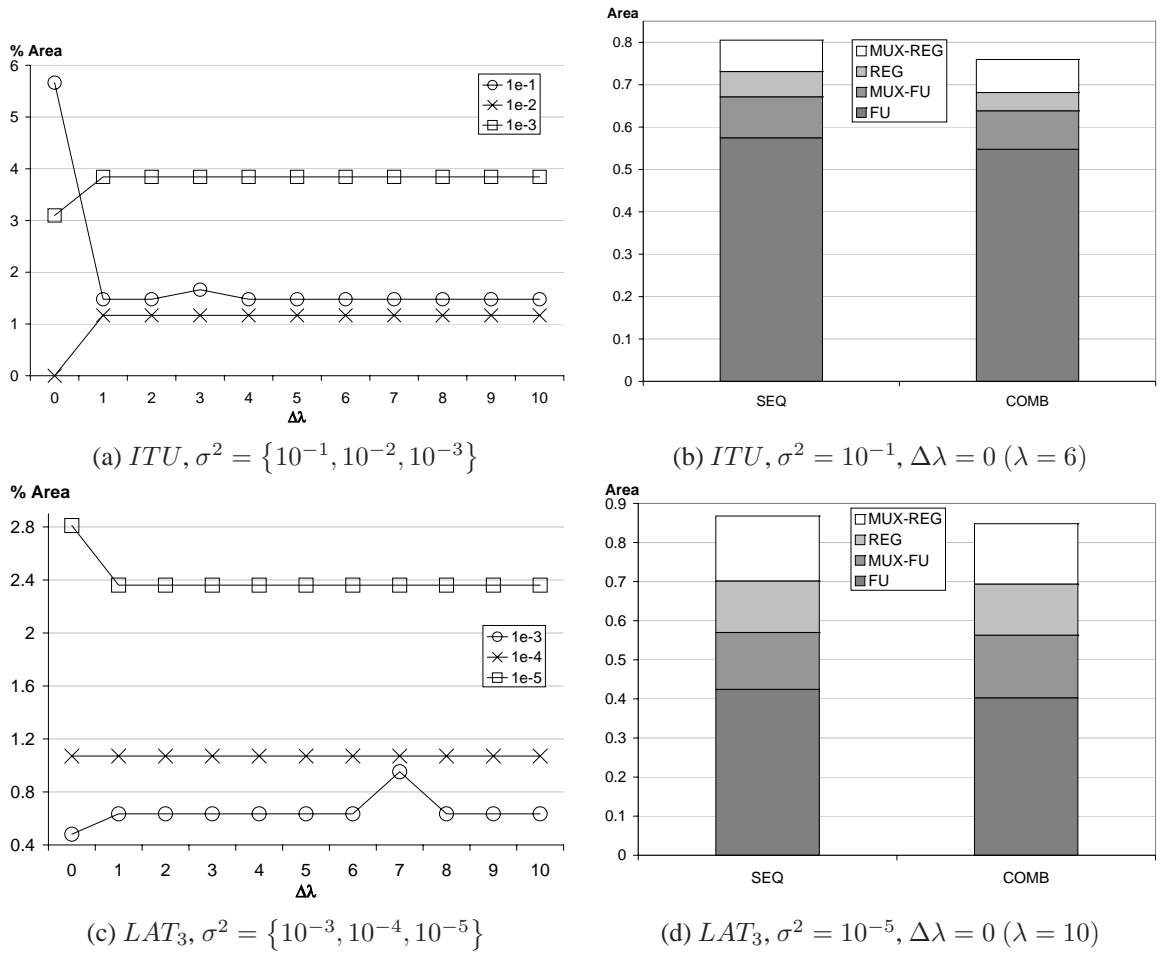
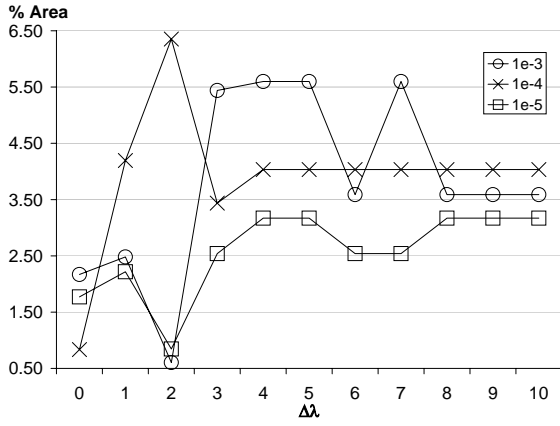


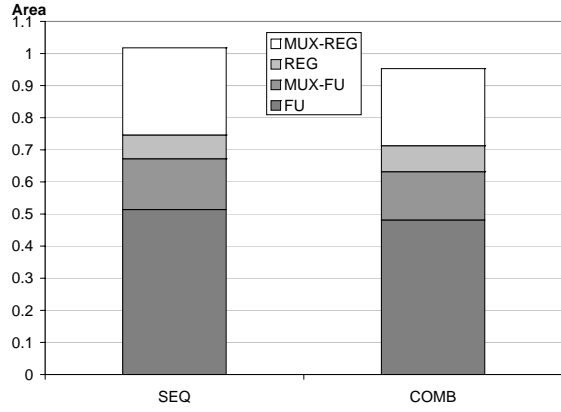
Figure 5.1: *SEQ* vs. *COMB*: homogeneous implementations (I)

(%) vs. $\Delta\lambda$ (latency displacement from the minimum latency) for three different output noise constraints (Fig. 5.1-a,-c, Fig. 5.2-a,-c, and Fig. 5.3-a,-c), while the right subfigures contain a detailed resource distribution graph of a particular point of its counterpart (Fig. 5.1-b,-d, Fig. 5.2-b,-d and Fig. 5.3-b,-d). The latency ranges from the minimum latency achievable by the MWL system ($\lambda_{min}^{MWL-HOM}$) to $\lambda_{min}^{MWL-HOM} + 10$. The latency of DCT_8 has been restricted to $\lambda = \lambda_{min}^{MWL-HOM} + \{3, 6, 9, 12\}$, due to the large computation time required.

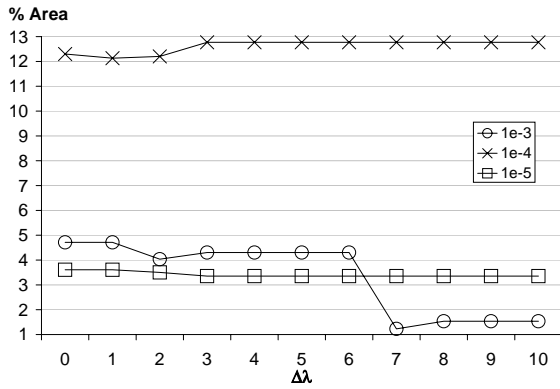
Fig. 5.1-a displays the area improvements of *ITU* obtained by *COMB* for three different noise constraints. The results show that for all cases area improvements are obtained. However, there are only a few noise/latency scenarios where the improvements are signif-



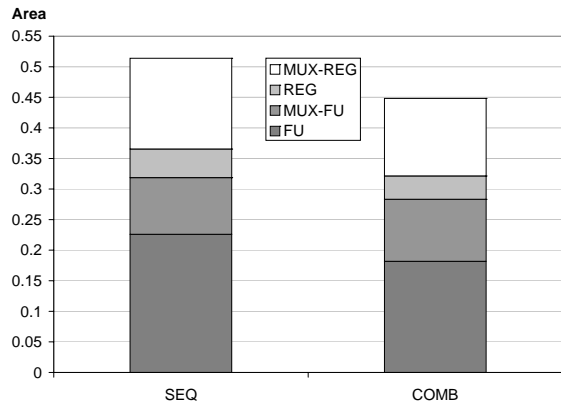
(a) $IIR_4, \sigma^2 = \{10^{-1}, 10^{-2}, 10^{-3}\}$



(b) $IIR_4, \sigma^2 = 10^{-4}, \Delta\lambda = 2 (\lambda = 8)$



(c) $FIR_8, \sigma^2 = \{10^{-3}, 10^{-4}, 10^{-5}\}$



(d) $FIR_8, \sigma^2 = 10^{-4}, \Delta\lambda = 3 (\lambda = 11)$

Figure 5.2: *SEQ* vs. *COMB*: homogeneous implementations (II)

icant ($\sigma^2 = 10^{-1}, \Delta\lambda = 0; \sigma^2 = 10^{-2}, \Delta\lambda = [1, 10]$). This situation recalls that of the optimal case obtained in the previous section, where improvements were sporadic. However, now there is an improvement for the majority of cases, yet small. This is due to the fact that the design space is now larger. Also, the latencies of resources are now dependent on the word-lengths, which adds new optimization possibilities.

Fig. 5.1-b displays the *ITU* detailed resource distribution for $\sigma^2 = 10^{-1}$ and $\Delta\lambda = 0$ ($\lambda = 6$). The improvement of 5.66% is mainly due to a decrease in the area of FUs' and registers simultaneously.

The analysis of Fig. 5.1-a,-c, Fig. 5.2-a,-c and Fig. 5.3-a,-c shows that, as expected from

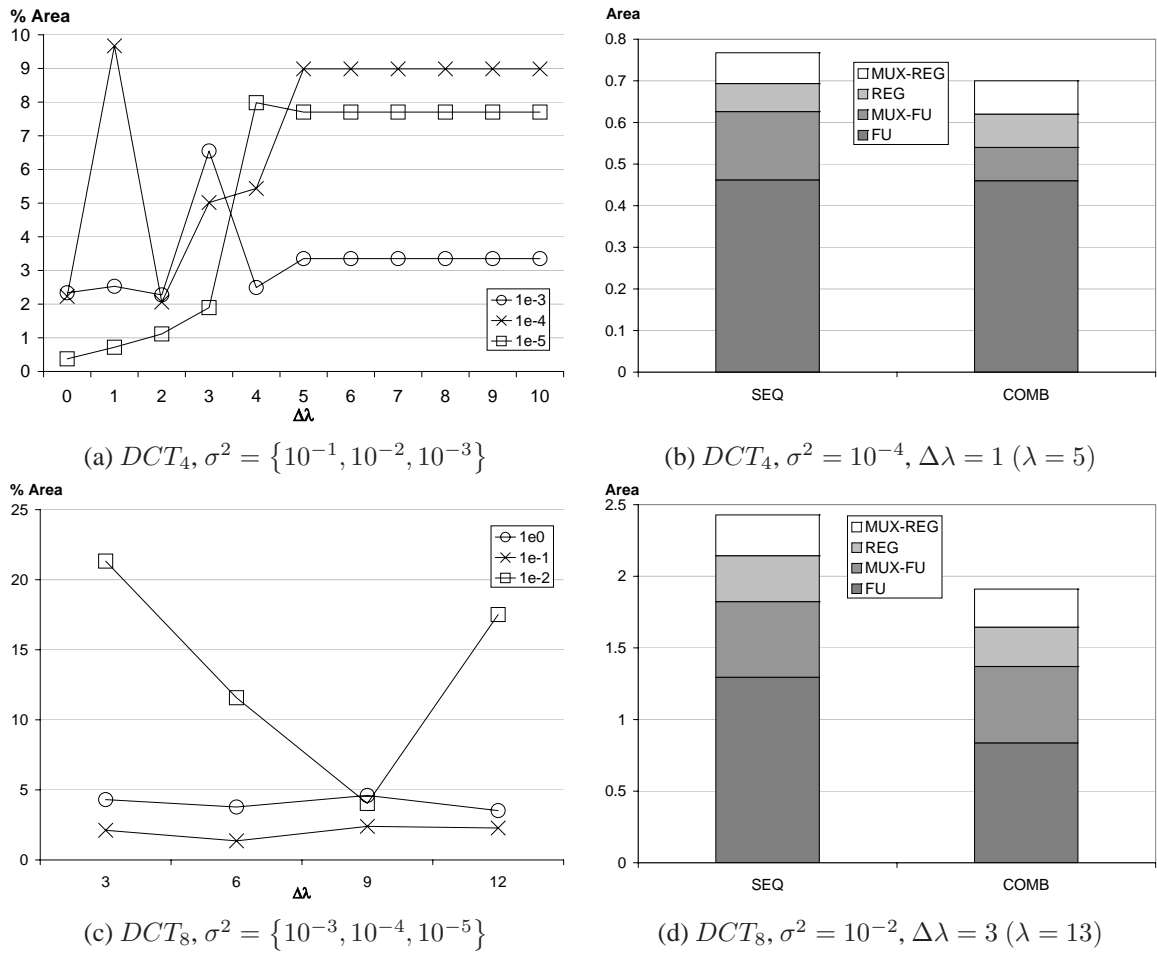


Figure 5.3: SEQ vs. COMB: homogeneous implementations (III)

the optimal analysis, the relationship between area improvement and λ or σ^2 is not clear. The ITU , LAT_3 and FIR_8 graphs show that latency has little effect on the improvements obtained, while for IIR_4 and DCT_4 the area improvement fluctuates as the latency increases. DCT_8 presents both cases. The erratic nature of the area improvements obtained suggests that there is no real relationship between the improvements and latency or noise.

The detailed resource distribution graphs provide with a variety of situations: the improvement in ITU (Fig. 5.1-b) is mainly obtained by a reduction in the area of FUs and registers; the improvements in LAT_3 (Figs. 5.1-d) are due to a reduction in FU's area, and there are both an increase of the area of FU's multiplexers and a decrease of the area of reg-

Table 5.4: *SEQ* vs. *COMB*: homogeneous implementations

Bench.	σ_q^2	Area improvement (%)		
		min	max	mean
<i>ITU</i>	10^{-1}	1.48	5.66	1.88
	10^{-2}	0.00	1.17	1.06
	10^{-3}	3.10	3.84	3.78
<i>LAT₃</i>	10^{-3}	0.48	0.95	0.65
	10^{-4}	1.07	1.07	1.07
	10^{-5}	1.82	2.36	2.31
<i>IIR₄</i>	10^{-3}	0.84	6.35	3.91
	10^{-4}	0.61	5.60	3.80
	10^{-5}	0.85	3.17	2.57
<i>FIR₈</i>	10^{-3}	1.23	4.71	3.32
	10^{-4}	12.13	12.77	12.62
	10^{-5}	3.35	3.61	3.41
<i>DCT₄</i>	10^{-1}	2.27	6.55	3.30
	10^{-2}	2.06	9.67	7.12
	10^{-3}	0.37	7.98	5.30
<i>DCT₈</i>	1	3.77	4.60	4.05
	10^{-1}	1.36	2.40	2.04
	10^{-2}	4.03	21.32	12.30
<i>All</i>		0.00	21.32	3.93

isters' multiplexers; in *IIR₄* and *FIR₈* (Fig. 5.2-b,-d) FUs and registers' multiplexers are reduced and *FIR₈* presents a simultaneous increase and decrease of the area of FUs' multiplexers and registers' multiplexers; the improvements in *DCT₄* and *DCT₈* (Fig. 5.3-b,-d) are due to a reduction in FUs' multiplexers and FUs, respectively.

Table 5.4 holds results on the homogeneous-resource implementations of all benchmarks corresponding to three different quantization noise scenarios. For each quantization scenario the latency ranges from $\lambda_{min}^{MWL-HOM}$ to $\lambda_{min}^{MWL-HOM} + 10$, except for *DCT₈*, and the minimum, maximum and mean values of the area improvements obtained by the *SEQ* implementations in comparison to the *COMB* implementations are computed. The first column

in the table contains the name of the benchmark. The second, the output noise variance applied. And, the third column contains the minimum, maximum and mean area improvement values. The last row holds the minimum, maximum and average improvements considering all results simultaneously.

Looking at the complexity of the benchmarks (table 5.3 and the at the results, the improvements suggest that there is a tendency to obtain better results as long as the complexity (i.e. number of operations) of the algorithms increases. Summarizing, for the majority of cases the improvements are non-zero, the mean improvement is of 3.93% and the maximum improvement achieved is 21.32%.

Figures 5.4, 5.5 and 5.6 contain results of the comparison of *SEQ* vs. *COMB* using an heterogeneous-resource architecture (i.e. both LUT-based and embedded resources present). The subfigures on the left show the area improvements ($\|\hat{\mathbf{A}}\|_{\infty}$, A_{LUT} and A_{EMB}) vs. latency results, while the subfigures on the right show the detailed resource distributions. The latency ranges from the minimum latency achievable by the MWL system ($\lambda_{min}^{MWL-HET}$) to $\lambda_{min}^{MWL-HET} + 10$.

Figures 5.4-a,-c, 5.5-a,-c and 5.5-a,-c, again display a variety of different behaviors. There are benchmarks with a virtually constant response in terms of ∞ -norm (Fig. 5.4-a,-c and 5.5-c), and for others the ∞ -norm varies highly from point to point. It is clear that the main ∞ -norm improvement is produced by a reduction of LUT-based resources. However, there are some cases (*ITU*, $\sigma^1 = 10^{-3}$, $\lambda = 6$; *DCT₄*, $\sigma^1 = 10^{-3}$, $\lambda = [8, 14]$) where there is no improvement at all in terms of ∞ -norm, but the LUT-based resources are greatly reduced (around 10%). There are also cases where the overall improvement comes with an increase in the original number of embedded resources (*DCT₈*, $\sigma^1 = 10^0$, $\lambda = 12$). *ITU*, $\sigma^1 = 10^{-3}$, $\lambda = 6$)

The detailed resource distribution graphs show that: *ITU* area is reduced 9% since the total area of FUs and FUs' multiplexers is reduced (note that FUs' area increases while mul-

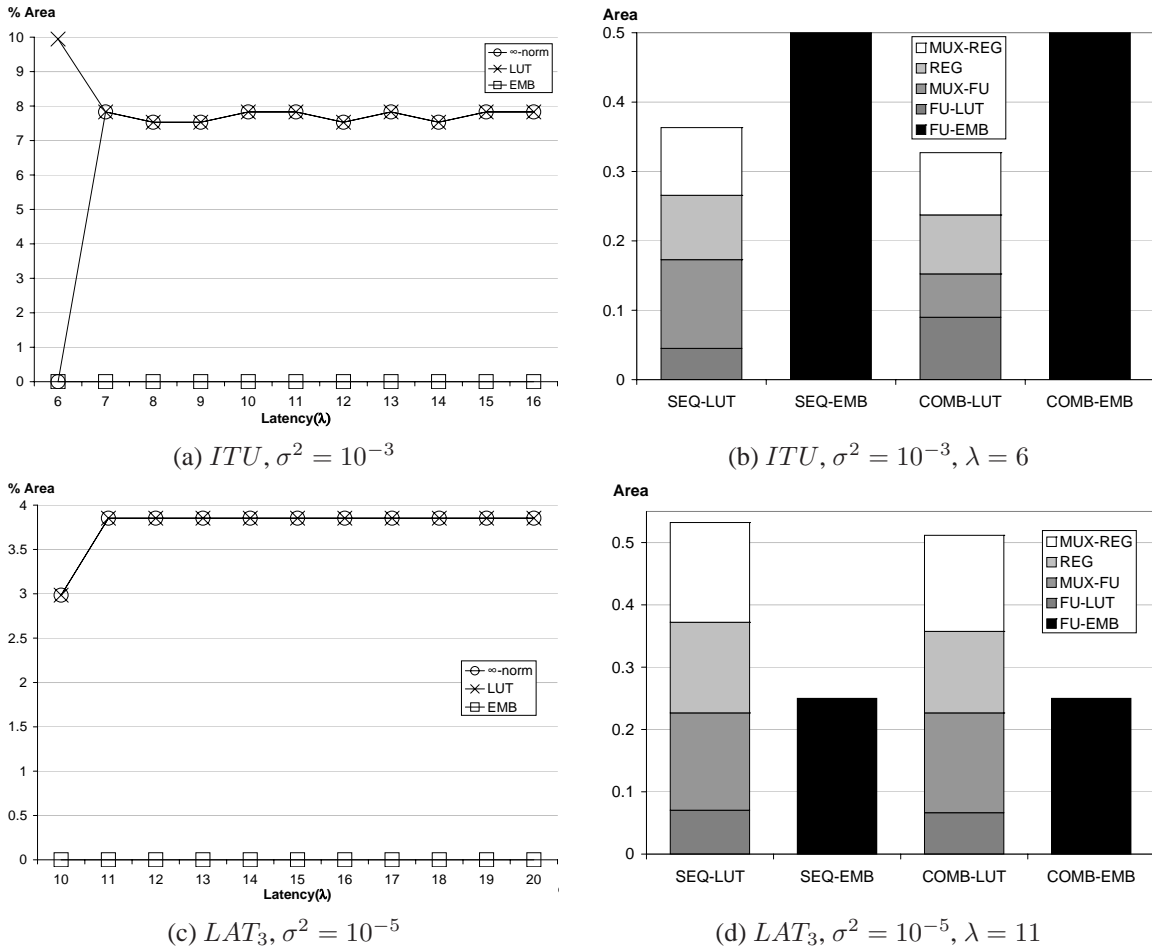


Figure 5.4: SEQ vs. COMB: heterogeneous implementations (I)

tiplexers' area decreases); LAT_3 achieves an area reduction of 8% mainly due to a reduction in the area of registers; IIR_4 presents an interesting case where the area of embedded resources is doubled to allow an 8% reduction of the area of LUT-based resources (mainly registers), resulting in an overall area reduction; FIR_8 has a 9.68% area reduction due to a decrease of registers and multiplexers; DCT_4 has no ∞ -norm improvement, though the area of FUs and registers is reduced. Finally, the dramatic area reduction (18%) produced in DCT_8 is basically due to a reduction of the area of FUs, although the areas of registers and multiplexers are also reduced.

Table 5.5 contains results for all benchmarks corresponding to heterogeneous-resource

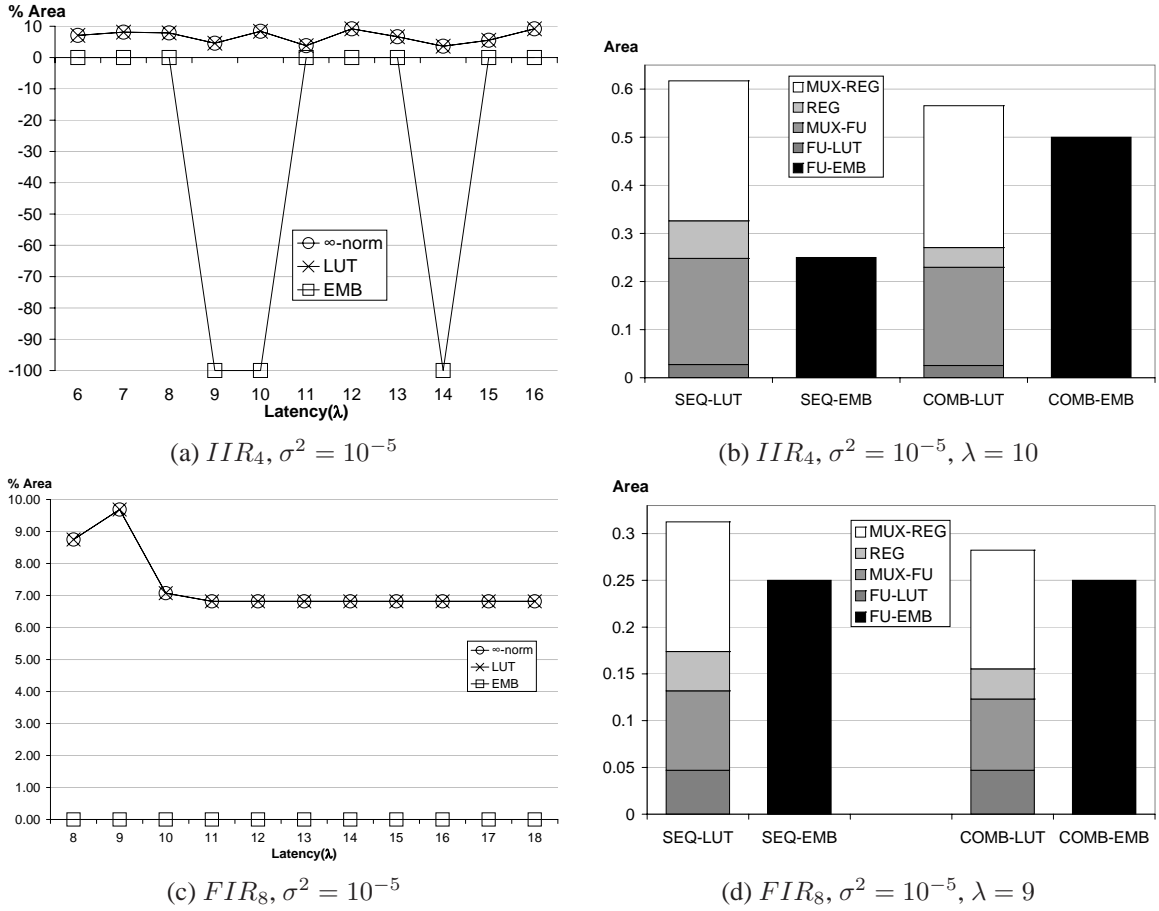
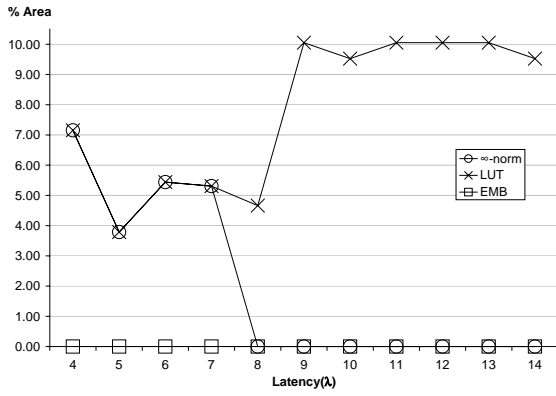
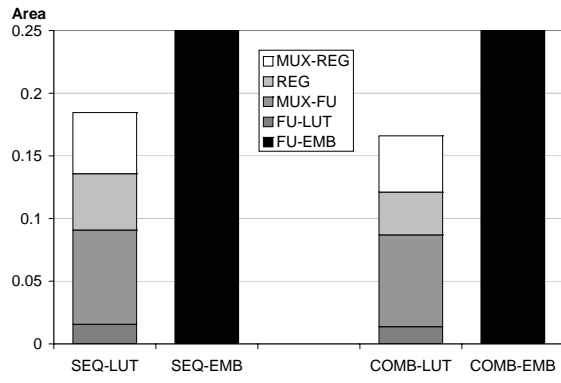


Figure 5.5: *SEQ* vs. *COMB*: heterogeneous implementations (II)

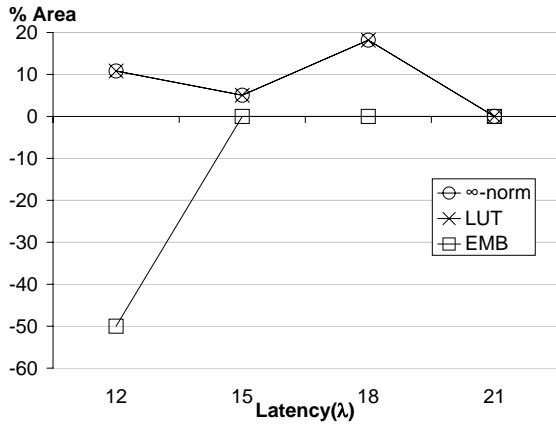
FPGAs. For each quantization scenario the latency ranges from $\lambda_{min}^{MWL-HET}$ to $\lambda_{min}^{MWL-HET} + 10$ (except DCT_8), and the minimum, maximum and mean values of the area improvements obtained by the *SEQ* implementations in comparison to the *COMB* implementations for three noise constraints are computed. The first column in the table contains the name of the benchmark. The second, the output noise variance applied. The third column contains the minimum, maximum and mean ∞ -norm area improvement values. The fourth column contains the minimum, maximum and mean area improvement obtained for the smaller resource component (i.e. LUT-based or embedded resources) when $\|\hat{\mathbf{A}}\|_{\infty}^{SEQ} = \|\hat{\mathbf{A}}\|_{\infty}^{COMB}$. Thus, if two implementations (*SEQ* and *COMB*) has the same



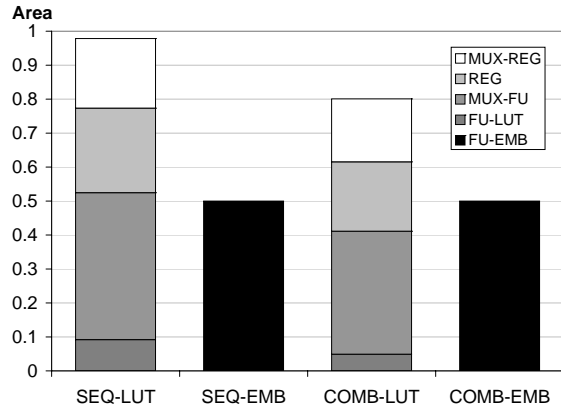
(a) $DCT_4, \sigma^2 = 10^{-3}$



(b) $DCT_4, \sigma^2 = 10^{-3}, \lambda = 9$



(c) $DCT_8, \sigma^2 = 10^0$



(d) $DCT_8, \sigma^2 = 10^0, \lambda = 18$

Figure 5.6: SEQ vs. COMB: heterogeneous implementations (III)

∞ -norm values, the information in this column indicates which one is wasting less resources. The last row holds the minimum, maximum and average improvements considering all results simultaneously.

The results show that there is an improvement in terms of ∞ -norm for the majority of cases. The mean improvement ranges from 0.0% to 14.42%, being bigger than 4% for ITU ($\sigma^2 = \{10^{-1}, 10^{-3}\}$), IIR_4 , FIR_8 ($\sigma^2 = \{10^{-4}, 10^{-5}\}$), DCT_4 ($\sigma^2 = \{10^{-1}, 10^{-3}\}$) and DCT_8 . The maximum improvement obtained is 18.16% for DCT_8 . As to the cases when $\|\hat{A}\|_{\infty}^{SEQ} = \|\hat{A}\|_{\infty}^{COMB}$ the resource improvement ranges from 0.0% to 10.05%, with a mean value of 5.60%. The improvements are bigger than 4% for ITU , IIR_4 and FIR_8 and DCT_4 .

Table 5.5: SEQ vs. COMB: heterogeneous implementations

Bench.	σ_q^2	$\ \hat{\mathbf{A}}\ _\infty$ %			$A_{LUT/EMB} \%$ $\ \hat{\mathbf{A}}\ _\infty^{SEQ} = \ \hat{\mathbf{A}}\ _\infty^{COMB}$		
		min	max	mean	min	max	mean
<i>ITU</i>	10^{-1}	0.00	4.56	0.41	3.17	3.57	3.25
	10^{-2}	0.00	1.78	1.62	0.00	0.00	0.00
	10^{-3}	0.00	7.83	7.01	9.95	9.95	9.95
<i>LAT₃</i>	10^{-3}	0.56	1.66	1.11	-	-	-
	10^{-4}	1.77	1.77	1.77	-	-	-
	10^{-5}	2.99	3.85	3.73	-	-	-
<i>IIR₄</i>	10^{-3}	0.00	8.81	4.89	4.25	7.63	5.68
	10^{-4}	6.78	8.36	7.90	-	-	-
	10^{-5}	3.64	9.19	6.47	-	-	-
<i>FIR₈</i>	10^{-3}	0.00	0.00	0.00	6.47	8.23	7.61
	10^{-4}	6.82	9.69	7.54	-	-	-
	10^{-5}	4.16	5.54	5.11	-	-	-
<i>DCT₄</i>	10^{-1}	0.00	7.15	3.09	4.66	10.05	9.13
	10^{-2}	0.00	1.83	0.40	0.00	7.09	5.00
	10^{-3}	0.00	6.00	1.78	0.00	0.00	0.00
<i>DCT₈</i>	1	0.00	18.16	11.34	-	-	-
	10^{-1}	12.32	13.72	12.91	-	-	-
	10^{-2}	12.49	14.42	14.42	-	-	-
<i>All</i>		0.00	18.16	4.48	0.00	10.05	5.60

Optimization times

Table 5.6 contains information regarding typical optimization times. The computer used to carry out the experiments has a 1.66 GHz Intel Core Duo processor and 1 GB of RAM. The latency constraint for these experiments was set to $\lambda = \lambda_{min} + 3$ clock cycles, while the noise constraint is indicated in the third column of the table. The last two columns show the optimization times required for the *SEQ* and the *COMB* approaches.

As expected, the combined approach requires longer times than the sequential approach. Also, it can be seen that as long as the complexity of the benchmarks increases the opti-

Table 5.6: Optimization times: *SEQ* vs. *COMB*.

Arch.	Bench.	σ^2	<i>SEQ</i> (secs)	<i>COMB</i> (secs)
HOM	<i>ITU</i>	10^{-1}	50.99	47.59
	<i>LAT</i> ₃	10^{-3}	177.93	350.40
	<i>IIR</i> ₄	10^{-3}	151.18	249.00
	<i>FIR</i> ₈	10^{-3}	115.55	295.20
	<i>DCT</i> ₄	10^{-3}	181.74	270.00
	<i>DCT</i> ₈	10^0	17513	30350
HET	<i>ITU</i>	10^{-1}	69.64	70.66
	<i>LAT</i> ₃	10^{-3}	197.33	317.10
	<i>IIR</i> ₄	10^{-3}	148.68	265.20
	<i>FIR</i> ₈	10^{-3}	203.67	267.60
	<i>DCT</i> ₄	10^{-3}	157.18	245.1
	<i>DCT</i> ₈	10^0	25573	42195

mization times also increase. It is interesting to see that the time increase is not linear. For instance, the ratio between the number of nodes (see table 5.3) and the optimization times between *DCT*₈ and *DCT*₄ for the sequential homogeneous case are 3.18 and 96.36, respectively; and for the combined homogeneous case are 3.18 and 112.4. This depicts how the complexity of the optimization process is highly affected by any increase in the algorithm complexity.

Summarizing, the results show that the joint application of WLA and HLS provides area improvements for the majority of experiment performed. In particular, less than 1% of the syntheses carried out (324 in total) presented no improvement. As expected from the optimal analysis, the advantage of the proposed approach arises sporadically, with improvements of up to 21% for homogeneous architectures and up to 18% ($\|\hat{\mathbf{A}}\|_{\infty}$) for heterogeneous architectures. The mean improvements are 3.93% and 4.48% respectively. Also, in heterogeneous implementations there is an overall resource improvement even when the ∞ -norm improvement is zero (maximum value of 10% and mean value of 5.60%). In general, area improve-

ments (∞ -norm) greater than 4% occur in 38% of the syntheses performed, and improvements greater than 7% in 26%.

5.3. Conclusions

In this chapter the combined application of WLA and HLS has been addressed. Both optimal and heuristic approaches have been studied.

The optimal approach has been implemented using mathematical programming (i.e. MILP). It has the following features:

- It provides with a compact mathematical description of the combined problem by means of MILP
- It addresses systems that comply with:
 - Linear Time-Invariant properties.
 - 1-cycle latency resources.
 - Only multipliers are shared.
 - Only FUs and registers due to delays are considered.
- It can be used to assess the quality of heuristic optimization techniques.
- The analysis of the results provides insight regarding the combined optimization process that may shed some light in the design of new combined WLA and HLS heuristics.

The optimal results show that:

- Improvements up to 13% are obtained
- The improvements are obtained sporadically only for particular combinations of latency and noise constraints.

- The analysis of the optimal results enables the proposal of several heuristic WLA and HLS optimization steps.

A novel combined WLA and HLS heuristic procedure has been presented. Its main features are:

- SA-based approach.
- Complete set of datapath resources that covers FUs, registers and multiplexers.
- MWL-oriented set of resources that includes word-length dependent resource cost models of area and latency.
- FPGA-oriented set of resources composed of both LUT-based and embedded resources.
- Simultaneous optimization of word-lengths and data-path architecture.
- Optimizer able to handle such a complex set of resources.
- Optimizer able to change the current solution by means of:
 - Movements based on HLS premises.
 - Movements based on WLA premises.
 - Specific movements to the combined WLA and HLS paradigm.

The results show that:

- The combined WLA and HLS approach provides area improvements in 99% of the experiments (324 in total).
- The area improvements are greater than 4.0% in 38% of the experiments.
- The area improvements are greater than 7.0% in 26% of the experiments.

- There are improvements of up to 21% with a mean value of 3.93% for homogeneous architectures.
- There are improvements of up to 18% with a mean value of 4.48% for heterogeneous architectures.
- The overall resource improvements are up to 10% with a mean value of 5.6% when $\|\hat{\mathbf{A}}\|_{\infty}^{SEQ} = \|\hat{\mathbf{A}}\|_{\infty}^{COMB}$ in heterogeneous architectures.

CHAPTER 6

CONCLUSIONS

6.1. Conclusions

This work has addressed the combined application of Word-Length Allocation and High-Level Synthesis of Digital Signal Processing circuits. These two design tasks are focused on the optimization of DSP circuits, therefore the interest of their joint application, since further optimization levels might be reached. WLA is aimed at the cost optimization (area, power and speed) by means of tailoring the precision of the arithmetic operations involved in the signal processing. The HLS of DSP circuits has a similar optimization goal, though it is focused on the selection of a crafted circuit architecture that minimizes resource waste. On one hand, HLS is highly dependent on WLA, since the number of bits assigned to signals determines the size of functional units, buses, etc. On the other hand, optimum WLA would require knowledge of the architecture (i.e. output of HLS). Thus, the interest in combining mathematical precision and circuit architecture exploration into a single design task. While pursuing the ultimate goal of this thesis, there have been contributions to the state of the art

that not only include those related to the combined approach, but that also cover the tasks of WLA and HLS separately.

The MWL HLS task has been dealt with in chapter 3, having as an extra goal to tune the results to modern DSP-oriented FPGAs. An SA-based approach has been adopted due to its resiliency and its ability to obtain quasi-optimal results. SA is also known for the long computation times required for convergence, but the point of view taken in this work is that of analyzing the possibilities of the MWL HLS approach (as well as the combined optimization) instead of focusing on its computational efficiency, which is seen as a future research phase. The cost modeling of resources (e.g. area and latency) has been a key factor when adding MWL functionality to the proposed HLS algorithm. While similar models are present in the literature, one of the main points of the model presented in this work is the simultaneous consideration of functional units, registers and multiplexers. A novel area lower bound for MWL multiplexers that accounts for input data word-lengths and alignment has been provided. The results show that the percentage of the total area dedicated to multiplexers and registers is usually high, exposing a lack of efficient structures for multiplexing in current FPGAs. Also, the use of a complete resource set is justified so the optimization process has a more complete architectural model than the traditional *only-FUs* model. Area improvements of up to 35% (mean 2%) have been reported in the comparison results between using a naive set of resources vs. a complete one. The architectural model is completed even further as embedded multipliers are also included. A novel optimization cost function that enables the automatic module selection required to efficiently distribute LUT-based and embedded resources has been proposed. The achieved overall resource usage minimization would have not been possible if existing minimization functions had been used. The UWL vs. MWL results obtained for the set of benchmarks show area improvements up to 77% (mean 46%) and average latency improvements of 22% when applying resource sharing in

homogeneous-architecture FPGAs. Heterogeneous-architecture implementations show area improvements up to 80 % (mean 44 %) and average latency improvements of 19%. The comparison of MWL homogeneous vs. MWL heterogeneous architectures yields that there are improvements up to 54% (mean 40%) when embedded resources are included in the optimization. In summary, the main contributions and conclusions regarding MWL HLS are:

- A complete tool to develop optimized MWL DSP circuits for homogeneous- and heterogeneous-architecture FPGAs is provided.
- The tool considers FUs, registers and multiplexers simultaneously.
- A novel area metric (+-norm) is used to handle both LUT-based and embedded resources.
- MWL clearly outperforms the traditional UWL approach.
 - Homogeneous architectures: average 46% improvement.
 - Heterogeneous architectures: average 44% improvement.
 - Minimum latency improvement: 22% (homogeneous) and 19% (heterogeneous)
- The use of efficient module selection techniques for heterogeneous systems achieves average area improvements of 40%.
- The datapath implementation results show that a high percentage of the resource usage is devoted to multiplexers and registers, exposing a deficiency in FPGA multiplexing structures.

The WLA of DSP systems has been tackled in chapter 4 focusing on two main points: the generation of a fast and accurate quantization error estimator, and the analysis of the effect

of different optimization WLA techniques on the HLS results. Quantization noise estimation is essential in order to perform complex – time-consuming – circuit optimizations, since the traditional simulation-based approach leads to exceedingly long optimization times, thus limiting the complexity of the optimization techniques that can be used. However, only LTI systems in steady state can be fully analyzed in order to obtain an accurate noise model. In this work, a noise estimator for LTI and a reduced set of non-linear systems (which includes recursive algorithms) has been presented. The estimation is based on the use of AA, since AA-based simulations provide with information of the effect of any perturbation (i.e. quantization noise) applied to a signal on the rest of the signals of the algorithm, keeping information about the origin of the perturbation. This enables the parameterization of the statistical properties of the output noise as a function of the statistical properties of the noise sources (i.e. signals). Since the goal of this thesis is to combine two already quite complex design tasks, leading to an even more complex approach, any speedup obtained is of most importance. Hence the stress put on the generation of a fast noise estimator. The novel estimator presented here achieves high accuracy for LTI and some non-feedback non-linear algorithms (average error $<1.21\%$ for $SQNR = [3, 120]$) and for feedback non-linear algorithms (average error $<3.64\%$ for $SQNR = [40, 120]$). The achieved time boost compared to bit-true simulations is up to 1210-fold. As for the use of different WLA optimization techniques, the results yield that the behavior obtained when the architecture is assumed to be non-sharing are not kept when resource sharing is allowed during HLS. For instance, if the architecture of the implementation is non-sharing, an SA-based WLA performs better than a gradient-descent approach. However, when applying WLA and HLS (with resource sharing) sequentially, in some cases there is no difference between the approaches and it is even possible that gradient-descent outperforms SA. This is due to the fact that the non-sharing architectural assumption, which is implicit during WLA is not correct, since it does not match the final resource-sharing architecture. The results show that applying WLA and

HLS sequentially introduces some uncertainty about the quality of WLA optimization methods, showing, again, the necessity of the combined approach. However, complex WLA techniques achieves better average improvements compared to the basic gradient-descent technique. For instance, SA achieves a mean area improvement of 5.18% in homogenous implementations, and of 4.75% in heterogeneous implementations. Summarizing, the main contributions and conclusions related to WLA are:

- A novel quantization noise estimator is presented:
 - Applicable to LTI and a subset of non-linear algorithms, including transients.
 - Highly accurate for LTI and non-feedback non-linear systems: mean error < 2%.
 - Highly accurate for feedback non-linear systems: mean error < 4%.
 - Estimation speedups up to $\times 1210$.
- Current WLA optimization methods do not account for architectural issues leading to sub-optimal results.
 - The performance of WLA after HLS is not guaranteed to be maintained (sub-optimal techniques might perform better than quasi-optimal techniques).
 - SA-based WLA obtains an average improvement of 5% compared to gradient-descent.

The combined application of WLA and HLS (chapter 5) has been addressed following two different approaches: an optimal approach and a heuristic approach. The optimal approach is handled by means of mathematical programming (MILP). Due to the extremely long convergence times required, the original problem has been simplified and applied to small benchmarks. The optimal combined approach has been compared to the sequential application of two separate MILP-based WLA and HLS. The results obtained shed some

light over the possible steps that an efficient combined WLA-HLS heuristic algorithm may take. The combined approach implies simultaneous variations in word-length sizes, the set of resources, as well as the binding between operation and resources. The optimal analysis results yield that improvements are obtained in a scattered manner, since only particular combinations of noise and latency constraints allow achieving significant differences when comparing to the sequential approach. FU's area improvements up to 13% are obtained. The MILP approach enabled the selection of several SA movements used for the heuristic combined approach. These movements comprise traditional resource binding movements, traditional word-lengths modifications, as well as specific movements that combine word-length and resource binding modifications. As a result, a synthesis algorithm able to optimize DSP implementations restricted to noise and latency constraints has been proposed. Again, the set of resources is composed of FUs, registers and multiplexers, and embedded resources have been also included. The approach has been compared to the sequential execution of separate SA-based WLA and HLS. The comparison yields that the combined approach produces an improvement in the majority of cases. The improvements for homogeneous implementations are around 4% with a maximum value of 21%. The heterogeneous implementations add a new dimension to the problem, since the improvement can be measured twofold: in terms of the number of times that the circuit can be replicated on the FPGA (inversely related to ∞ -norm), or in terms of the overall resource reduction when the ∞ -norms are equivalent. The number of times that the circuit can be implemented is improved around 4.5% with a maximum value of 18%. The overall resource usage has a mean improvement of 5.6% with a maximum at 10%. There is a tendency that suggests that improvements increases as long as the complexity of the benchmarks increases. As previously mentioned, the next phase in the research is aimed at optimizing the efficiency of the tools, so complex algorithms can be synthesized in feasible times, in order to confirm the improvements provided by the combined approach in industrial cases. The main contributions and conclusions are:

- An MILP description of the problem is provided.
- The optimal results show that improvements appear in a scattered manner, with a maximum of 13%. They are used to extract possible heuristic strategies.
- A complete tool to develop optimized MWL DSP circuits for homogeneous- and heterogeneous-architecture FPGAs able to simultaneously select the word-lengths of signals and the architecture of the implementation.
- The tool considers FUs, registers and multiplexers.
- The tool considers both LUT-based and embedded resources.
- The combined approach produces benefits for most cases.
- Homogeneous implementations are improved around 4% (maximum 21%).
- Heterogeneous architectures present:
 - a mean ∞ -norm area improvement of 4.5% (maximum 18%).
 - a mean overall resource usage improvement of 5.6% (maximum 10%).
- The results suggests that the improvements obtained increase along with the DSP algorithm complexity.

6.2. Future research lines

During the analysis of the combination of WLA and HLS it has been necessary to contribute to the state of the art in these two fields separately, since the current state of these design tasks did not allowed a seamless integration. Thus, this thesis has not only studied the combination of two well known tasks, but it has also improved the quality of current WLA and HLS design techniques separately. This made the research work of a considerable magnitude and some issues had to be left outside this study.

For instance, the HLS of MWL systems has still many open issues. MWL register/memory binding is still to be further analyzed. Configurable devices demand for efficient techniques to map variables to LUT-based register, LUT-based memory and embedded memory, within the MWL paradigm. Furthermore, new FPGA architectures demands the inclusion of complex DSP blocks as well as hierarchical embedded memories within the resource set. The \pm -norm is to be applied to this sort of optimization problems. Another important issue is that of multiplexers optimization (multiplexer allocation and port assignments). An MWL approach to multiplexer optimization can highly benefit from the traditional UWL approach. Moreover, this issue turns essential when using FPGAs, due to the waste of resources produced by multiplexers in datapath implementations. In addition, it is of most interest to apply the proposed MWL HLS to overlapped scheduling [DM94, WP92], since this may enable the implementation of faster DSP systems that still make use of resource sharing as a means to reduce cost.

Also, quantization noise estimation seems to be a promising research line to follow. The generation of fast estimators opens the door to more thorough and faster optimization techniques. The estimator proposed in this work can be extended to consider higher-degree Taylor expansion models that allow the parameterization of non-linear operations with increased accuracy. Also, it can be oriented to consider other error metrics, such as bit or symbol error

rates (BER,SER) targeted at communication systems [CSB04].

Computation efficiency must also be addressed by means of generating fast heuristics for the combined problem. Operation grouping seems an interesting choice [HMS05]. The grouping can be performed considering the quantization noise parameters – again, the need for quantization noise estimators – and data dependencies and temporal scheduling. Also, clock selection could be addressed in order to obtain better implementations. Works such as [Cho01] could be oriented to the combined problem. Of course, the complexity of clock selection would be increased, since the latency of the resources may change throughout the optimization process. Therefore, as aforementioned, efforts have to be made to improve the efficiency of the tool implementation. As a final remark, power optimization could be of great interest and works such as [JC08, CGC06] can be adopted to the proposed combined approach. An integral power optimization techniques that accounts for the power consumption of FUs, multiplexers, buses, registers and embedded resources could highly benefit from a combined approach.

Finally, and as a complementary line to the latter, the integration of the WLA and HLS algorithms proposed within commercial HLS frameworks is another interesting line to follow. HLS commercial tools, such as Catapult [Cat] or Impulse-C [Imp], could be extended by means of a combined approach. A possibility is to perform an optimization loop where the design constraints and the very code describing the algorithm are modified iteratively. It would be necessary to perform a study on the impact on the final implementation results that the constraints and the code style used produce. This research could provide with some clues about how to improve commercial tools by means of the combination of WLA and HLS.

BIBLIOGRAPHY

- [Ach93] H. Achatz. Extended 0/1 LP Formulation for the Scheduling Problem in High-Level Synthesis. In *Proc. Design Automation Conference*, pages 226–231, 1993.
- [Alt] Altera Corp. www.altera.com.
- [AO04] H.A. Atat and I. Ouais. Register Binding for FPGAs with Embedded Memory. In *Proc. IEEE Symp.on Field-Programmable Custom Computing Machines*, pages 165–175, 2004.
- [BCC05] C.-S. Bouganis, G.A. Constantinides, and P.Y.K. Cheung. A Novel 2D Filter Design Methodology for Heterogeneous Devices. In *Proc. IEEE Symposium on Field-Programmable Custom Computing Machines*, 2005.
- [BFS04] C. Brandolese, W. Fornaciari, and F. Salice. An Area Estimation Methodology for FPGA Based Designs at SystemC-Level. In *Proc. Design Automation Conference, 2004*, pages 129–132, 2004.
- [BGP00] S. Bilavarn, G. Gogniat, and J.L. Philippe. Area Time Power Estimation for FPGA Based Designs at a Behavioral Level. In *Proc. Int. Conf. on Electronics, Circuits and Systems*, volume 1, pages 524–527, 2000.
- [BM00] Luca Benini and Giovanni de Micheli. System-Level Power Optimization: Techniques and Tools. *Proc. ACM Trans. Des. Autom. Electron. Syst.*, 5(2):115–192, 2000.
- [BMU92] N. Benvenuto, M. Marchesi, and A. Uncini. Applications of simulated annealing for the design of special digital filters. *IEEE Trans. on Signal Processing*, 40(2):323–332, Feb 1992.

- [BP00] A. Benedetti and P. Perona. Bit-Width Optimization for Configurable DSPs by Multi-interval Analysis. In *Proc. IEEE Asilomar Conf. on Signals, Systems and Computers*, 2000.
- [BR05] P. Belanovic and M. Rupp. Automated Floating-point to Fixed-point Conversion with the fixify Environment. In *Proc. Int. Workshop on Rapid System Prototyping*, pages 172–178, June 2005.
- [Cam90] R. Camposano. From Behavior to Structure: High-Level Synthesis. *IEEE Des. Test. Comput.*, 7(5):8–19, 1990.
- [Cat] Catapult Synthesis - Mentor Graphics. www.mentor.com.
- [CC94] D. Chen and J. Cong. Register Binding and Port Assignment for Multiplexer Optimization. In *Proc. IEEE Asilomar Conf. on Signals, Systems and Computers*, volume 1, pages 68–73, 1994.
- [CCC⁺06] G. Caffarena, G.A. Constantinides, P.Y.K. Cheung, C. Carreras, and O. Nieto-Taladriz. Optimal Combined Word-Length Allocation and Architectural Synthesis of Digital Signal Processing Circuits. *IEEE Trans. Circuits Syst. II*, 53(5):339–343, May 2006.
- [CCL99] G.A. Constantinides, P.Y.K. Cheung, and W. Luk. Truncation Noise in Fixed-Point SFGs. *IEE Electronics Letters*, 35(23):2012–2014, 1999.
- [CCL00a] G.A. Constantinides, P.Y.K. Cheung, and W. Luk. Multiple-Wordlength Resource Binding. In *Proc. Int. Conf. on Field-Programmable Logic and Applications*, volume 1896, 2000.
- [CCL00b] G.A. Constantinides, P.Y.K. Cheung, and W. Luk. Optimal Datapath Allocation for Multiple-Wordlength Systems. *IEE Electronics Letters*, 36(17):1508–1509, 2000.
- [CCL03a] G. A. Constantinides, P. Y. K. Cheung, and W. Luk. Wordlength Optimization for Linear Digital Signal Processing. *IEEE Trans. Computer-Aided Design*, 22(10):1432–1442, October 2003.
- [CCL03b] G.A. Constantinides, P.Y.K. Cheung, and W. Luk. Synthesis of Saturation Arithmetic Architectures. *ACM Trans. on Design Automation of Electronics Systems*, 8(3):334–354, 2003.
- [CCL05] G.A. Constantinides, P.Y.K. Cheung, and W. Luk. Optimum and Heuristic Synthesis of Multiple Word-Length Architectures. *IEEE Trans. VLSI Syst.*, 13(1):39–57, 2005.
- [CCR01] A. B. Carlson, P.B. Crilly, and J. C. Rutledge. *Communication Systems*. Prentice-Hall, New York, 2001.

- [CFC⁺05] G. Caffarena, A. Fernandez, C. Carreras, J.A. López, and O. Nieto-Taladriz. Design of a High-Speed WLAN Adaptive Equalizer: Implementation Issues. *Med. J. of Electron. and Commun.*, 1(1):25–34, October 2005.
- [CFCNT04a] G. Caffarena, A. Fernandez, C. Carreras, and O. Nieto-Taladriz. Fixed-Point Refinement of OFDM-Based Adaptive Equalizers: A Heuristic Approach. In *Proc. European Signal Processing Conference EUSIPCO'04*, pages 1353–1356, 2004.
- [CFCNT04b] G. Caffarena, A. Fernandez, C. Carreras, and O. Nieto-Taladriz. Implementation of an RLS Equalizer for OFDM-based WLAN. In *Proc. Int. Symposium on Circuits, Systems and Networks for Digital Signal Processing*, pages 604–607, 2004.
- [CFH⁺05] J. Cong, Yiping Fan, Guoling Han, Yizhou Lin, Junjuan Xu, Zhiru Zhang, and Xu Cheng. Bitwidth-Aware Scheduling and Binding in High-Level Synthesis. In *Proc. Asian South Pacific Design Automation Conference*, 2005.
- [CGC05] J.A. Clarke, A.A. Gaffar, and G.A. Constantinides. Parameterized Logic Power Consumption Models for FPGA-based Arithmetic. In *Proc. Int. Conf. on Field Programmable Logic and Applications*, pages 626 – 629, 2005.
- [CGC06] J.A. Clarke, A.A. Gaffar, and G.A. Constantinides. Fast Word-Level Power Models for Synthesis of FPGA-Based Arithmetic. In *Proc. IEEE Int. Symp. on Circuits and Systems*, pages 1299–1302, 2006.
- [CH02] M.L. Chang and S. Hauck. Precis: A Design-Time Precision Analysis Tool. In *Proc. IEEE Symp. on Field-Programmable Custom Computing Machines*, pages 229–238, 2002.
- [CH05] M.L. Chang and S. Hauck. Précis: A Usercentric Word-Length Optimization Tool. *IEEE Des. Test. Comput.*, 22(4):349–361, July 2005.
- [Cho01] Sanghun Park; Kiyoungh Choi. Performance-Driven High-Level Synthesis with Bit-Level Chaining and Clock Selection. *IEEE J. Technol. Computer Aided Design*, 20(2):199–212, 2001.
- [CJH96] Jeong-Il Choi, Hong-Shin Jun, and Sun-Young Hwang. Efficient Hardware Optimisation Algorithm for Fixed Point Digital Signal Processing ASIC Design. *IEE Electronics Letters*, 32(11):992–994, 1996.
- [CLCNT06a] G. Caffarena, J. A. López, C. Carreras, and O. Nieto-Taladriz. High-Level Synthesis of Multiple Word-Length DSP Algorithms using Heterogeneous-Resource FPGAs. In *Proc. Field Programmable Logic and Applications*, pages 675–678, 2006.

- [CLCNT06b] G. Caffarena, J. A. López, C. Carreras, and O. Nieto-Taladriz. Optimized Implementation of DSP Cores on FPGAs Using Logic-based and Embedded Resources. In *Symp. on System-on-Chip*, pages 103–106, 2006.
- [CLNT99] Carlos Carreras, Juan A. López, and Octavio Nieto-Taladriz. Bit-Width Selection for Data-Path Implementations. In *Proc. Int. Symp. on System synthesis*, page 114, 1999.
- [Con03] G.A. Constantinides. Perturbation Analysis for Word-Length Optimization. In *Proc. IEEE Symp. on Field-Programmable Custom Computing Machines*, pages 81–90, 2003.
- [CRS⁺99] R. Cmar, L. Rijnders, P. Schaumont, S. Vernalde, and I. Bolsens. A Methodology and Design Environment for DSP ASIC Fixed Point Refinement. In *Proc. Conf. on Design, Automation and test in Europe*, page 56, 1999.
- [CSB04] R.W. Changchun Shi; Brodersen. Floating-Point to Fixed-Point Conversion with Decision Errors due to Quantization. *Proc. IEEE Int. Conf. on Acoustics, Speech, and Signal Processing*, 5:41–44, 2004.
- [CSL02] M.-A. Cantin, Y. Savaria, and P. Lavoie. A Comparison of Automatic Word Length Optimization Procedures. In *IEEE Int. Symposium on Circuits and Systems*, volume 2, pages 612–615, 2002.
- [CSPL01] M.-A. Cantin, Y. Savaria, D. Prodanos, and P. Lavoie. An Automatic Word Length Determination Method. In *IEEE Int. Symp. on Circuits and Systems*, volume 5, pages 53–56 vol. 5, 2001.
- [CVDM88] F. Catthoor, J. Vandewalle, and H. De Man. Simulated Annealing Based Optimization of Coefficient and Data Word-Lengths in Digital Filters. *J. Circuit Theory Applications*, 16(1):371–390, September 1988.
- [CWP95] Yun-Nan Chang, Ching-Yi Wang, and K.K. Parhi. DSP Synthesis with Heterogeneous Functional Units Using the MARS-II System. In *Proc. IEEE Asilomar Conf. on Signals, Systems and Computers*, volume 1, pages 109–116 vol.1, 1995.
- [DM94] Giovanni De Michelli. *Synthesis and Optimization of Digital Circuits*. Series in Electrical and Computer Engineering. McGraw-Hill, New York, 1994.
- [EJCT00] Rolf Enzler, Tobias Jeger, Didier Cottet, and Gerhard Tröster. High-level area and performance estimation of hardware building blocks on fpgas. In *Proc. Field-Programmable Logic and Applications*, pages 525–534, 2000.
- [FCR03] C.F. Fang, Tsuhan Chen, and R.A. Rutenbar. Floating-Point Error Analysis Based on Affine Arithmetic. In *Proc. IEEE Int. Conf. on Acoustics, Speech, and Signal Processing*, volume 2, pages 561–564, 2003.

- [GCC06] A.A. Gaffar, J.A. Clarke, and G.A. Constantinides. PowerBit - Power Aware Arithmetic Bit-Width Optimization. In *Proc. IEEE Int. Conf. on Field Programmable Technology*, pages 289–292, 2006.
- [GDWL92] D. Gajski, N. Dutt, A. Wu, and S. Lin. *High-Level Synthesis*. Kluwer Academic, Boston, 1992.
- [GML⁺02] A.A. Gaffar, O. Mencer, W. Luk, P.Y.K. Cheung, and N. Shirazi. Floating-Point Bitwidth Analysis Via Automatic Differentiation. In *IEEE Int. Conf. on Field-Programmable Technology*, pages 158–165, 2002.
- [GML04] A.A. Gaffar, O. Mencer, and W. Luk. Unifying Bit-Width Optimisation for Fixed-Point and Floating-Point Designs. In *Proc. IEEE Symp. on Field-Programmable Custom Computing Machines*, pages 79–88, 2004.
- [GN72] R. S. Garfinkel and G. L. Nemhauser. *Integer Programming*. Wiley, New York, 1972.
- [Hay02] S. Haykin. *Adaptive Filter Theory*. Prentice-Hall, Upper Saddle River, 2002.
- [Hay03] B. Hayes. A Lucid Interval. *American Scientist*, 91(6):484–488, 2003.
- [HE06] Kyungtae Han and B.L. Evans. Optimum Wordlength Search Using Sensivity Information. *EURASIP Journal of Applied Signal Processing*, 2006:1–14, 2006. doi:10.1155/ASP/2006/92849.
- [HES04] K. Han, B.L. Evans, and E.E. Swartzlander. Data Wordlength Reduction for Low-Power Signal Processing Software. In *Proc. IEEE Workshop on Signal Processing Systems*, pages 343–348, 2004.
- [HKBR06] M. Holzer, B. Knerr, P. Belanovic, and M. Rupp. Efficient design methods for embedded communication systems. *EURASIP Journal on Embedded Systems*, (ID 64913), 2006.
- [HMS05] N. Herve, D. Menard, and O. Sentieys. Data Wordlength Optimization for FPGA Synthesis. In *Proc. IEEE Workshop on Signal Processing Systems*, pages 623–628, 2005.
- [Imp] Impulse C - Impulse Accelerated Technologies. www.impulsec.com.
- [JC07] R. Jevtic and G. Carreras, C. Caffarena. Switching Activity Models for Power Estimation in FPGA Multipliers. In *Proc. Int. Workshop on Applied Reconfigurable Computing*, pages 201–213, 2007.
- [JC08] R. Jevtic and G. Carreras, C. Caffarena. Fast and Accurate Power Estimation of FPGA DSP Components Based on High-level Switching Activity Models. *Int. Journal of Electronics*, 2008. to be published.

- [JCC07] R. Jevtic, C. Carreras, and G. Caffarena. High-level Switching Activity Models for Multipliers in FPGAs. In *Proc. ACM/SIGDA Int. Symp. on Field Programmable Gate Arrays*, pages 224–225. ACM Press, 2007.
- [KGV83] S. Kirkpatrick, C.D. Gelatt, and M.P. VecchiKim. Optimization by Simulated Annealing. *Science*, 220(4598):671–680, 1983.
- [KKS95] Seehyun Kim, Ki-II Kum, and Wonyong Sung. Fixed-Point Simulation Utility for C and C++ Based Digital Signal Processing Programs. In *Proc. IEEE Workshop on VLSI Signal Processing*, pages 197–206, 1995.
- [KKS98] S. Kim, K.-I. Kum, and W. Sung. Fixed-Point Optimization Utility for C and C++ Based Digital Signal Processing Programs. *IEEE Trans. Circuits Syst. II*, 45(11):1455–1464, November 1998.
- [KKS00] K.I. Kum, J. Kang, and W. Sung. AUTOSCALER for C: An Optimizing Floating-Point to Integer C Program Converter for Fixed-Point Digital Signal Processors. *IEEE Trans. Circuits Syst. II*, 47(9):840–848, September 2000.
- [KS01] K. I. Kum and W. Sung. Combined Word-Length Optimization and High-Level Synthesis of Digital Signal Processing Systems. *IEEE Trans. Circuits Syst.*, 20(8):921–930, August 2001.
- [LCCNT03a] J.A. Lopez, G. Caffarena, C. Carreras, and O. Nieto-Taladriz. Characterization of the Quantization Properties of Similarity-Related DSP Structures by Means of Interval Simulations. In *Proc. IEEE Asilomar Conf. on Signals, Systems and Computers*, volume 2, pages 2208–2212, 2003.
- [LCCNT03b] J.A. Lopez, C. Carreras, G. Caffarena, and O. Nieto-Taladriz. Fast Characterization of the Noise Bounds Derived from Coefficient and Signal Quantization. In *Proc. IEEE Int. Symp. on Circuits and Systems*, volume 4, pages 309–312, 2003.
- [LCCNT04] J.A. Lopez, G. Caffarena, C. Carreras, and O. Nieto-Taladriz. Analysis of Limit Cycles by Means of Affine Arithmetic Computer-aided Tests. In *Proc. European Signal Processing Conference EUSIPCO'04*, pages 991–994, 2004.
- [LCNT07] J.A. Lopez, C. Carreras, and O. Nieto-Taladriz. Improved Interval-Based Characterization of Fixed-Point LTI Systems With Feedback Loops. *IEEE Trans. Computer-Aided Design*, 26(11):1923–1933, November 2007.
- [LGC⁺06] D.-U. Lee, A.A. Gaffar, R.C.C. Cheung, W. Mencer, O. Luk, and G.A. Constantinides. Accuracy-Guaranteed Bit-Width Optimization. *IEEE Trans. Computer-Aided Design*, 25(10):1990–2000, October 2006.
- [Lin97] Y-L Lin. Recent Developments in High-Level Synthesis. *ACM Trans. on Design Automation of Electronic Systems*, 2(1):2–21, 1997.

- [Liu71] B. Liu. Effect of Finite Word Length on the Accuracy of Digital Filters – A Review. *IEEE Trans. Circuits Syst.*, 18(6):670–677, 1971.
- [LMD94] B. Landwehr, P. Marwedel, and R. Dömer. OSCAR: Optimum Simultaneous Scheduling, Allocation and Resource Binding Based on Integer Programming. In *Proc. European Conf. on Design Automation*, pages 90–95, 1994.
- [Lóp04] J.A. López. *Evaluación de los Efectos de Cuantificación en las Estructuras de Filtros Digitales Mediante Técnicas de Simulación Basadas en Extensiones de Intervalos*. PhD thesis, Universidad Politécnica de Madrid, 2004.
- [Lue84] D. G. Luenberger. *Linear and Non-Linear Programming*. Addison-Wesley, Reading, Mass., 1984.
- [LV99] M. López-Vallejo. *Métodos para la Distribución de Funcionalidad entre Recursos Hardware y Software en el Diseño de Sistemas Heterogeneos*. PhD thesis, Universidad Politécnica de Madrid, 1999.
- [LVSB05] X. Liang, J.S. Vetter, M.C. Smith, and A.S. Bland. Balancing FPGA Resource Utilities. In *Proc. Int. Conf. on Eng. of Reconf. Systems and Algorithms*, pages 156–162, 2005.
- [Mat] Matlab - The Mathworks Inc. www.mathworks.com.
- [MCC05] G.W. Morris, G.A. Constantinides, and P.Y.K. Cheung. Using DSP Blocks for ROM Replacement: A Novel Synthesis Flow . In *Proc. Int. Conf. Field Programmable Logic and Applications*, pages 77–82, 2005.
- [MN05] P. Metzgen and D. Nancekievill. Multiplexer Restructuring for FPGA Implementation Cost Reduction. In *Proc. Design Automation Conference*, pages 421 – 426, 2005.
- [MOS] MOSEK ApS. www.mosek.com.
- [MPC90] M.C. McFarland, A.C. Parker, and R. Camposano. The High-Level Synthesis of Digital Systems. *Proceedings of the IEEE*, 78(2):301–318, 1990.
- [MRS⁺01] S. Mahlke, R. Ravindran, M. Schlansker, R. Schreiber, and T. Sherwood. Bitwidth Cognizant Architecture Synthesis of Custom Hardware Accelerators. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 20(11):1355–1371, 2001.
- [MRSMH06] M.C. Molina, R. Ruiz-Sautua, J.M. Mendias, and R. Hermida. Bitwise Scheduling to Balance the Computational Cost of Behavioral Specifications. *IEEE Trans. Computer-Aided Design*, 25(1):31–46, 2006.

- [MRSS04] D. Menard, R. Rocher, P. Scalart, and O. Sentieys. SQNR Determination in Non-Linear and Non-Recursive Fixed-Point Systems. In *Proc. European Signal Processing Conference EUSIPCO'04*, pages 1349–1352, 2004.
- [MS02a] D. Menard and O. Sentieys. A Methodology for Evaluating the Precision of Fixed-Point Systems. In *IEEE International Conference on Acoustics, Speech, and Signal Processing, 2002. Proceedings.*, volume 3, pages 3152–3155, 2002.
- [MS02b] D. Menard and O. Sentieys. Automatic Evaluation of the Accuracy of Fixed-Point Algorithms. In *Proc. Design, Automation and Test in Europe Conference and Exhibition*, pages 529–535, 2002.
- [NHCB02] A. Nayak, M. Haldar, A. Choudhary, and P. Banerjee. Accurate Area and Delay Estimators for FPGAs. In *Proc. Design, Automation and Test in Europe Conference and Exhibition, 2002*.
- [OKD97] Seong Yong Ohm, F.J. Kurdahi, and N.D. Dutt. A Unified Lower Bound Estimation Technique for High-Level Synthesis. *IEEE Trans. Computer-Aided Design*, 16(5):458–472, May 1997.
- [OKSH06] H. Orsila, T. Kangas, E. Salminen, and T.D. Hamalainen. Parameterizing Simulated Annealing for Distributing Task Graphs on Multiprocessor SoCs. In *Proc. IEEE Int. Symp. on System-on-Chip*, pages 1–4, 2006.
- [OS87] A. V. Oppenheim and R. W. Schaffer. *Discrete-Time Signal Processing*. Prentice-Hall, Englewood Cliffs, NJ, 1987.
- [OW72] Alan V. Oppenheim and Clifford J. Weinstein. Effects of Finite Register Length in Digital Filtering and the Fast Fourier Transform. *Proceedings of the IEEE*, 60(8):957–976, 1972.
- [Par99] Keshab K. Parhi. *VLSI Digital Signal Processing Systems: Design and Implementation*. Wiley, New York, 1999.
- [PK89] P.G. Paulin and J.P. Knight. Force-Directed Scheduling for the Behavioral Synthesis of ASICs. *IEEE Trans. Computer-Aided Design*, 8(6):661–679, 1989.
- [RB05] S. Roy and P. Banerjee. An Algorithm for Trading Off Quantization Error with Hardware Resources for MATLAB-Based FPGA Design. *IEEE Trans. Comput.*, 54:886–896, 2005.
- [RM87] R. A. Roberts and C. T. Mullis. *Digital Signal Processing*. Addison-Wesley, Reading, Mass., 1987.

- [RMHS06] R. Rocher, D. Menard, N. Herve, and O. Sentieys. Fixed-Point Configurable Hardware Components. *EURASIP Journal on Embedded Systems*, 2006:Article ID 23197, 13 pages, 2006. doi:10.1155/ES/2006/23197.
- [SB04a] Changchun Shi and R.W. Brodersen. A Perturbation Theory on Statistical Quantization Effects in Fixed-Point DSP with Non-Stationary Inputs. In *Proc. IEEE Int. Conf. on Circuits and Systems*, volume 3, pages 373–376 Vol.3, 2004.
- [SB04b] Changchun Shi and R.W. Brodersen. Automated Fixed-Point Data-Type Optimization Tool for Signal Processing and Communication Systems. In *Proc. Design Automation Conference*, pages 478–483, 2004.
- [SF97] J. Stolfi and L. H. Figueiredo. Self-Validated Numerical Methods and Applications. In *Brazilian Mathematics Colloquium: IMPA*, 1997.
- [SGDM93] K. Schoofs, G. Goossens, and H. De Man. Bit-Alignment in Hardware Allocation for Multiplexed DSP Architectures. In *Proc. European Conf. on Design Automation*, pages 289–293, 1993.
- [SK95] Wonyong Sung and Ki-Il Kum. Simulation-Based Word-Length Optimization Method for Fixed-Point Digital Signal Processing Systems. *IEEE Trans. Signal Processing*, 43(12):3087–3090, 1995.
- [Syn] Synplicity Inc. www.synplicity.com.
- [Wil02] S.J.E. Wilton. Implementing Logic in FPGA Memory Arrays: Heterogeneous Memory Architectures . In *Proc. IEEE Int. Conf. on Field-Programmable Technology*, 2002.
- [WKGM97] M. Willems, H. Keding, T. Grötzer, and H. Meyr. FRIDGE: An Interactive Fixed-Point Code Generation Environment for Hw/Sw-Codesign. In *Proc. Int. Conf. Acoustics, Speed, Signal Processing*, 1997.
- [WP92] C.-Y. Wang and K.K. Parhi. High Level DSP Synthesis Using the MARS Design System. In *Proc. IEEE Int. Symp. on Circuits and Systems*, volume 1, pages 164–167 vol.1, 1992.
- [WP95] Ching-Yi Wang and K.K. Parhi. High-Level DSP Synthesis Using Concurrent Transformations, Scheduling, and Allocation. *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, 14(3):274–295, 1995.
- [WP98] S.A. Wadekar and A.C. Parker. Accuracy Sensitive Word-Length Selection for Algorithm Optimization. In *Proc. Int. Conf. on Comp. Design*, pages 54–61, 1998.
- [Xil] Xilinx Inc. <http://www.xilinx.com>.

