RIGA TECHNICAL UNIVERSITY

Gatis VALTERS

FPGA IMPLEMENTATION OF PARAMETRICAL ORTHOGONAL TRANSFORM-BASED EXPERIMENTAL DSP DEVICES

Thesis synopsis

Riga 2011

RIGA TECHNICAL UNIVERSITY
Faculty of Electronics and Telecomunications
Institute of Radio Electronics

Gatis VALTERS
Student of the Doctoral study program "Radioelectronics"

FPGA IMPLEMENTATION OF PARAMETRICAL ORTHOGONAL TRANSFORM-BASED EXPERIMENTAL DSP DEVICES

Thesis synopsis

Academic supervisor
Dr. sc. ing., professor
P. MISĀNS

Riga 2011

UDK 621.391:517.44(043)
Va 434 p

Valters. G. FPGA Implementation of Parametrical Orthogonal Transform-based experimental DSP devices. Thesis synopsis. -R:RTU, 2011, -96 lpp.

Printed according to decision of Department of Fundamentals of Electronics, protocol No 1 of 7 September 2011.

This work has been supported by the European Social Fund within the project "Support for the implementation of doctoral studies at Riga Technical University".

ISBN 978-9934-10-262-2

DOCTORAL THESIS
FOR PROMOTION TO THE DOCTOR'S DEGREE IN ENGINEERING AT THE RIGA TECHNICAL UNIVERSITY

The public defense of the doctoral thesis for promotion to the doctor's degree in engineering will take place on 12 of January, 2012, at 15:45 in Room 210, Faculty of Electronics and Telecommunications, Azenes 12.

OFFICIAL REVIEWERS

Professor, Dr.sc.ing. Guntars Balodis
Faculty of Electronics and Telecommunications, Riga Technical University

Professor, Ph.D., Dipl.Eng. Peeter Ellervee
Tallin University of Technology, Department of Computer Engineering

Dr.sc.comp. Eugene Boole
Institute of Electronics and Computer Science, Senior Researcher

CONFIRMATION

I confirm that this doctoral thesis, submitted for a degree in engineering at the Riga Technical University, is my own work. The doctoral thesis has not been submitted for a degree in any other university.

Gatis Valters ..... (Signature)

Date: .....

The doctoral thesis is presented as a thesis by publication.

Contents

Abbreviations . . . 8
Nomenclature . . . 10
The Form of the Doctoral Thesis . . . 11
Significance of the Subject Matter . . . 11
Objectives . . . 13
Methods of Study . . . 13
Scientific Novelty and Main Results . . . 13
Defendable Theses . . . 14
Practical Significance of Research . . . 15
Approbation . . . 15
Thesis Contents . . . 19

Introduction **20**

1 Complex Parametrical Orthogonal Transforms **21**
1.1 Orthogonal, orthonormal and unitary transforms . . . 21
1.2 Rotation-angle-based unitary transforms . . . 21
1.2.1 Elementary generalized unitary rotation matrix . . . 21
1.2.2 Full unitary matrix . . . 24
1.2.3 Haar-like complex orthogonal transforms . . . 28
Summary . . . 32

2 Implementation Architectures for Orthogonal Transforms **34**
2.1 Parallel structure of transform implementation . . . 34
2.2 Serial structure of transform implementation . . . 34
2.2.1 **CRAIMOT** *BF* generator [P1], [P4] . . . 34
2.2.2 **CRAIMOT** spectrum analyzer [P2] . . . 38
Summary on papers [P1], [P2], [P4] . . . 39
2.3 Tree-like structure of signal decomposition/reconstruction system . . . 39
2.3.1 Signal decomposition [P10],[P11] . . . 40
2.3.2 Signal reconstruction . . . 45
Summary . . . 45
2.4 Signal decomposition using complex *FIR* filters . . . 46
2.4.1 2.4.1. Transfer functions of complex filters . . . 46
2.4.2 Cascading of complex orthogonal filters . . . 49
Summary . . . 52

3 FPGA Implementation of the Phi-Transform Algorithms 53
3.1 Floating- and fixed-point arithmetic . . . 53
3.2 *FPGA* implementation of *DSP* elements . . . 53
3.2.1 Multiplier with variable output wordlengths (*WL*) . . . 53
3.2.2 Serial-to-parallel and parallel-to-serial converters . . . 54
Summary . . . 56
3.3 Implementation of the generalized Jacobi rotator . . . 56
3.3.1 Traditional implementation using multipliers and adders [P9]-[P12] . 56
3.3.2 Implementation using *CORDIC* algorithm . . . 58
Summary on papers [P5], [P6], [P9]-[P12] . . . 60
3.4 Simplifications of the elementary generalized complex rotation matrix . . . 61
Summary on papers [P9], [P11] un [P12] . . . 62
3.5 Fixed-point error . . . 63
Summary . . . 65
3.6 Automation of implementation of *EGU*-rotator [P11], [P12] . . . 66
3.6.1 Design automation steps . . . 67
3.6.2 Automation tools . . . 69
Summary on papers [P11] un [P12] . . . 72

4 Experimental Devices 74
4.1 *FPGA* implementation of experimental **CRAIMOT** function generator . . 74
Summary on papers [P2] and [P5] . . . 76
4.2 *FPGA* implementation of experimental signal spectrum analyzer . . . 78
4.2.1 Signal spectrum analyzer with serial architecture [P2] . . . 78
4.2.2 *RE*-based signal spectrum analyzer-synthesizer [P5] . . . 79
Summary . . . 79
4.3 Signal analyzer-synthesizer . . . 80
Summary . . . 81
4.4 Haar-Like filters . . . 82
Summary on papers [P3], [P6] and [P8] . . . 85
4.5 The prototype-simulator of data transmission system based on the generalized orthogonal nonsinusoidal division multiplexing . . . 86
Summary on papers [P10] . . . 88

Conclusion 90

References 92

List of Figures

1.1 Decomposition of two signal samples 24
1.2 **CCRAIMOT** basis functions, $N = 8$, $\phi = [60°, 60°, 60°]$, $\psi = [60°, -60°, -60°]$, $\gamma = [60°, 60°, -60°]$ 27
1.3 Graphical depiction of permutation matrices for Haar-like transforms ($N = 8$) 30
1.4 **CRA-HT** basis functions, $N = 8$, $\phi = [30°, 30°, 30°]$, $\psi = [-30°, 30°, -30°]$, $\gamma = [-30°, 30°, -30°]$ 30
1.5 Trace of *BFs* vector end projections on the spread of surface of one of the unit sphere 3-D projections [P5] 32
2.1 Simplified block diagram of the **CRAIMOT** *BF* generator [P1] 36
2.2 Simplified block diagram of the **CRAIMOT** signal synthesizer 37
2.3 Optimized linear interpolation of sine for $\Delta_{max} \approx 0.004$ [P4] 38
2.4 Simplified block diagram of the **CRAIMOT** spectrum analyzer [P2] 39
2.5 **CCRAIMOT** *DE* (for N=4) simplified block diagram [P11] 40
2.6 Simplified timing diagram [P11] 40
2.7 Decomposition tree structure and addressing example $N = 8$ [P11] 41
2.8 First stage of decomposition of 4-sample signal 41
2.9 Decomposition of 4-sample signal 43
2.10 Graphical depiction of permutation matrices matrices ($N = 8$) 43
2.11 Simplified block diagram of **CCRAIMOT** reconstruction, $N = 4$ 45
2.12 Magnitude- and Phase-frequency response curves of the approximation and detail filters (**De**compozition), $\phi = \frac{\pi}{4}$, $\psi = \frac{\pi}{3}$, $\gamma = \frac{\pi}{4}$ 47
2.13 Magnitude- and Phase-frequency response curves of the approximation and detail filters (**Re**construction), $\phi = \frac{\pi}{4}$, $\psi = \frac{\pi}{3}$, $\gamma = \frac{\pi}{4}$ 48
2.14 De–Re filters 48
2.15 Magnitude response curves of the approximation and detail filters (**De**composition), taken partly ((a)) from [P3], [P6] un [P8]) 49
2.16 Simplified *Simulink* model of complex *FIR* filter 49
2.17 The **RA-HT** *DeRe* filter block diagram in *Simulink* [P3] 50
2.18 Simulink model for obtaining the complex matrix $\mathbf{T}_U(\phi, \psi, \gamma, 424)$ (filter coefficients) 50
2.19 *Simulink* model of the cascading of decomposition stages, using two *FIR* filters 51
2.20 *Simulink* timing diagrams of input signal decomposition, $[\phi = \frac{\pi}{4}, \psi = \frac{\pi}{3}, \gamma = \frac{\pi}{6}]$ 51
2.21 *Simulink* model of signal reconstruction 52
3.1 *RTL* description of multipliers using bit truncation, generated by *HDL Coder* 54
3.2 Serial-to-parallel converter (*Simulink* block diagram) 54
3.3 Timing diagram of the serial-to-parallel converter 55

3.4 Parallel-to-serial converter (*Simulink* block diagram) 55
3.5 Timing diagram of the parallel-to-serial converter 55
3.6 *RTL* description of the Jacobi rotator (no bit truncation) 57
3.7 *RTL* description of the Y_{Re} component calculation, using *fix* rounding ($w_{in} = 8$, $w_{mult} = 12$, $w_{out} = 8$) 58
3.8 Simplified block diagram of serial *EGURM*-rotator for (ϕ, $\psi = \frac{\pi}{2}$, $\gamma = 0$, ID=424) [P9] 62
3.9 Fixed-point errors for a 250-samples long noise-like signal 63
3.10 Distribution of *FPA* errors ($w_{in} = 8$, $w_{mult} = 8$, $w_{out} = 8$, *floor* rounding) . 64
3.11 Fixed-point *MSE*, for different *WL* (*floor* rounding) 65
3.12 Simplified design automation flow chart [P12] 67
3.13 Rotation Matrix Viewer *Graphical User Intarface GUI* [P12] 69
3.14 *Spectrum Expressions GUI* [P12] 70
3.15 *Simulink* model for *EGU*-rotator *VHDL* code generation [P12] 71
3.16 *HDLcoderGUI Graphical User Interface* 71
4.1 **CRAIMOT** *BF* generator implemented using *Quartus II 6.0* graphical environment 74
4.2 Simulation timing diagram for **CRAIMOT** *BF* generator (from *Quartus II 6.0*) [P1] 75
4.3 Shapes of ideal *BF* and experimental *BF* for $p = 6$ and $N = 16$ (the shape of captured *BF* is shifted up slightly for better comparison) [P1] 75
4.4 Simplified block diagram of **CRAIMOT** *BF* generator with parallel architecture [P4] 76
4.5 LCD snapshots [P2] 78
4.6 Simplified block diagram of the **RA-HT** spectrum analyzer [P5] 79
4.7 Example of speech signal compression 81
4.8 Extraction of single 2nd **CRA-HT** *BF* from noise (*SIMULINK* diagram) [P3] 83
4.9 Extraction of single **CRA-HT** *BF* from additive noise [P3] 83
4.10 Filtering of sine wave corrupted by pulse [P6] 84
4.11 Simplified test scheme of digital part of **CCRAIMOT** *GONDM* [P10] . . . 87
4.12 Simplified block diagram of *GONDM* data transmission system [P10] 87
4.13 Comparison of *BER* performances for different transforms and *AWGN* channel [P10] 88

Abbreviations

Abbreviation	Full phrase
FIR	Finite Impulse Respone
IIR	Infinite Impulse Response
RMS	Root Mean Square
MSE	Mean square error
HDL	Hardware description language
VHDL	Very high speed integrated circuits Hardware Description Language
FPA	Fixed Point Arithmetic
DSP	Digital Signal Processing
LUT	Look Up Table
ALUT	Adaptive Look Up Table
CORDIC	COordinate Rotation DIgital Computer
RTL	Register Transfer Level
TPD	Time Propagation Delay
SMT	Symbolic Math Toolbox
ASCII	American Standard Code for Information Interchange
BF	Basis Function
ADC	Analog-Digital Converter
DAC	Digital-Analog Converter
UC	Up Converter
DC	Down Converter
QAM	Quadrature Amplitude Modulation
QPSK	Quadrature Phase-Shift Keying
IEEE	Institute of Electrical and Electronics Engineers
OFDM	Orthogonal Frequency-Division Multiplexing
EGURM	Elementary Generalized Unitary Rotation Matrix
LE	Logic Element
FPGA	Field Programmable Gate Array
FT	Fast Transform
FFT	Fast Fourier Transform
IFFT	Inverse Fast Fourier Transform
DCT	Discrete Cosine Transform
AWGN	Additive White Gaussian Noise
FFC	Flat Fading Channel

FSFC	Frequency Selective Fading Channel
GONDM	Generalized Orthogonal Nonsinusoidal Division Multiplexing
M	Multipliers
A	Adders
MC	Multiplications by constant
SOPC	System On a Programmable Chip
EGURIT	Elementary Generalized Unitary Rotation Implementation Tool
UNITIT	UNItary Transform Implementation Tool
SANSYN	Spectrum ANalyzer-SYNthesizer

Nomenclature

Nomenclature	Explanation
Φ_k	Parameters collection (ϕ_k, ψ_k, γ_k)
$\mathbf{\Phi}$	Parameters matrix
$\mathbf{\Phi}_p$	Parameters matrix p−th column
$\mathbf{\Phi}_H$	Haar-like transform parameters matrix
$\mathbf{T}_U$	Generalized Jacobi rotation matrix
$\mathbf{B}_b$	Sparse factorized block-like rotation matrix
$\mathbf{B}_s$	Sparse factorized stair-like rotation matrix
$\mathbf{P}$	Permutation matrix
$\mathbf{0}$	Zeros matrix
$\mathbf{I}$	Identity matrix
$\mathbf{H}$	Orthogonal transform matrix
Φ	Orthogonal transform matrix (in older papers)
$\mathbf{H}_H$	Haar-like transform matrix
$(...)^{-1}$	Hermitian matrix
$\mathbf{X}, \mathbf{Y}$	Input and output signal vectors
$\delta_{\mathbf{p},\mathbf{k}}$	Kronecker symbol
c	$\cos(\phi)$
s	$\sin(\phi)$
N	Size of Transform matrix ($N = 2^n$)
n	Number of factorized matrices ($n = \log_2(N)$)
$\overline{(...)}$	Complex conjugate
$H(z), H(\omega)$	Filters transfer function
ω	angular frequency
ϵ_{RMS}	Root mean square error
$\widehat{\mathbf{v}}$	Signal vector with fixed point error
Fs	Sampling frequency
Ts	Sample time
w	*FPA* word length
Lo	Approximation filter
Hi	Detalization filter

The Form of the Doctoral Thesis

The doctoral thesis is presented in form of the author's scientific papers published from 2007 to 2011 and a review on these papers. There are many examples available of doctoral theses of such a kind (a review of published papers covering a single subject, e.g. [1]). For that reason it was not easy to choose in which form to submit the work. Firstly, because of the size – for the review part it varies from 15 to 200 pages, secondly, because no precise guidelines can be found in regulatory documents (listed in [2]). Therefore the following form, resembling also a standard doctoral thesis, has been chosen:

- the first part is like an introduction part of standard thesis synopsis
- the description part may be considered either like a shortened form of standard thesis or an extended body of standard thesis synopsis
- copies of published papers concerning the defendable work are included as appendices

The description part is written so that reader can understand the presented material without detailed examination of the publications. Most of the figures, tables and formulas are from the publications, but there are also additions that have either arisen after publication time or not been included in papers because of their restricted size.

Significance of the Subject Matter

This doctoral thesis deals with *FPGA* implementation issues of **CRABOT**[1] (complex rotation-angle-based, further named also as "angular") transforms. To avoid bulky designation, these transforms are named as "phi-transforms" [P8]. As Māris Tērauds stated in his doctoral thesis [3], "angles as parameters are not a novelty – they are used in Givens rotation, in *SVD*[2] [4], [5] and *EVD*[3] [6] algorithms." The mentioned thesis deals with real angular transforms (**RABOT** and its subclasses). Lately, as evidenced by the growing research in complex wavelets [7], research on complex angular transforms is also becoming more active. Over the past few decades, complex rotations (Jacobi rotations) have been hugely used in *SVD* and *QR*-algorithms (in eigenvalue calculations). With the rapid growth of complexity and performance of *ASIC* and *FPGA* circuits in the past decade, it has become more feasible to implement these algorithms into chips.

For the last six years, at the Faculty of Electronics and Telecommunications, Riga Technical University, under supervision of Professor P. Misāns theoretical and practical research is being done, concerning possible applications of angular transforms and, particularly, the use of these transforms for signal analysis/synthesis, signal compression, filtering, data transmission, image processing [P8]. Although two subclasses (**CRAOT**[4] and

[1] *Constant Rotation Angle Based Orthogonal Transform*
[2] *Singular Value Decomposition*
[3] *Eigenvalue Decomposition*
[4] *Constant Rotation Angle Orthogonal Transform*

CRAIMOT[5]) of **RABOT** transforms has been firstly described by Andrews – named as generalized Kronecker matrices [13], the RTU team has made a significant contribution in this field, by developing a methodology for the definition of novel parametrical transforms and creating various experimental devices, both virtual and *FPGA*-based. Researched transforms are parametrical transforms, characterized by several parameters (rotation angles). The parametrical representation of transforms allows to obtain orthogonal basis functions with an indefinite variety of forms. That, in its turn, extends possibilities of using digital signal processing in analysis/synthesis, compression, filtering, data transmission and allows to compete with well-known wavelets transforms [P8].

Historically, besides H.C.Andrews, A.M.Trahtman [9] also may be regarded as the founder of generalization of unitary transforms. Trahtman extended significantly the range of transforms used for analysis/synthesis of discrete signals, but he did not emphasize the use of rotation angles as parameters. Before the introduction of wavelets, the most significant contribution in generalization of transforms has been made by B.Fino and R.Algazi [10]. They formulated several rules for synthesis of unitary transforms, but again, with no emphasis on rotation angles. The first who introduced rotation-angle-based orthogonal filters is P.P.Vaidyanathan [11], but he did not use rotation angles in unitary transforms. Among wavelet researchers, definitely, must be mentioned P.Rieder with colleagues [12], who introduced parameterization of wavelets using rotation angles and used it in the *CORDIC* algorithm to implement wavelet filters. Without doubt, the use of novel transforms for signal analysis/synthesis and filtering opens up new opportunities, and research in this field is therefore very important. The same applies to the use of generalized frequency division multiplexing in data transmission systems, probably firstly introduced in [52].

The complexity and implementation of researched transforms are practically the same as in the case of fast Fourier transform [9] or wavelets, but the area of potential application is much more broad. On the one hand, for investigation of these application possibilities, it is important to diversify research directions (signal analysis/synthesis, signal compression, filtering, data transmission, image processing, etc. On the other hand, for assessment of hardware implementation possibilities for phi-transforms, it is important to diversify implementation of experimental devices. Besides, a deeper research is needed for hardware implementation issues of elementary generalized unitary rotation matrix – the core of angular transforms. The current development level of microelectronics allows to implement complex algorithms into hardware much easier than it was, for example, ten years ago. It considerably facilitates and, as a result, makes important, also the implementation in *FPGA* of rotation-angle-based signal processing algorithms. Since latest *FPGA* chips contain billions of transistors, automation of implementation of signal processing algorithms becomes more important. *FPGA* microchips, unlike *ASIC*, are more attractive in

[5] *Constant Rotation Angle Inside Matrix Orthogonal Transform*

the sense that they allow to maximally minimize the time for device creation and testing. Since *FPGA* families may have different architectures, performance, power consumption and the number of logic elements, then it is really necessary to estimate these parameters, depending on the required precision of signal processing.

Objectives

The thesis is a study on the *FPGA* implementation of the algorithms for **RABOT** and **CRABOT** transforms (phi-transforms). The main goal of work is **to deal with the problems related to a practical implementation of those algorithms and to conduct an approbation of the algorithms in *FPGA***. To accomplish this goal the following basic tasks have been stated:

1. Describe the elementary generalized unitary rotation matrix and its modifications.
2. Investigate and approbate the possibilities to automate the implementation of elementary generalized unitary rotation in *FPGA*.
3. Investigate and approbate the suitability of various architectures of rotation-angle-based basis function generators for implementation in *FPGA*.
4. Investigate and approbate the suitability of various architectures of rotation-angle-based spectrum analyzers for implementation in *FPGA*.
5. Investigate and approbate the possibilities to implement rotation-angle-based parametrical orthogonal filters in *FPGA*.

Methods of Study

The research process can be divided into four characteristic phases: analytical calculations, numerical calculations, simulation, and practical experiments. At first, analytical methods are used to prepare algorithms for subsequent implementation. It means manual using of matrix algebra and well-known properties of matrices, and using *MatLab* facilities for symbolic mathematical calculations (analytical calculations and inferences). Prototyping of algorithms is performed using programming in *MatLab* and *VHDL*, building models in *Simulink* environment, using numerical calculations and computer simulation with *Quartus II/ModelSim* software. Trial implementation of algorithms and testing of device prototypes are performed using Alteras *Quartus II* software and development kits.

Scientific Novelty and Main Results

The doctoral thesis studies, for the first time, *FPGA* implementation issues for rotation-angle-based fast unitary transforms (**CRABOT**). The main novelties are related to im-

plementation of algorithms for these transforms as experimental *FPGA*-based devices and programs:

- For the first time trial versions of parametrical *BF* generators, parametrical spectrum analyzers, parametrical orthogonal filters, based on the proposed algorithms and architectures, have been developed.
- For the first time a prototype in *FPGA* for the simulator of generalized orthogonal nonsinusoidal division multiplexing, based on the proposed algorithm and architecture, has been developed.
- For the first time a system for automated synthesis in *FPGA* of elementary generalized unitary rotation element, based on the proposed algorithm, has been developed.

The main results are:

- Definition of elementary generalized unitary rotation matrix.
- System based on *MatLab/Simulink/QuartusII/ModelSim* for automated synthesis in *FPGA* of elementary generalized unitary rotation element.
- Trial versions of parametrical orthogonal filters, that with an *FPA* precision perform the extraction/rejection of orthogonal signals.
- Prototype in *FPGA* for the simulator of generalized orthogonal nonsinusoidal division multiplexing. The parametrical **CCRAIMOT** transform is used.
- Trial versions of parametrical **CRAIMOT** *BF* generator.
- Trial versions of parametrical **CRAIMOT** and **RA-HT** spectrum analyzers.

Defendable Theses

The following theses are to be defended:

1. The developed system for automated synthesis of elementary generalized unitary rotation element (rotator), based on the proposed algorithm, essentially simplifies *FPGA*-based synthesis process – by decreasing the time required for synthesis and the consumption of logic elements, and by increasing operation speed.
2. The developed *FPGA*-based parametrical *BF* generators, parametrical spectrum analyzers and parametrical orthogonal filters, with serial, parallel and tree-like architectures, allow to choose most appropriate solutions, depending on operation speed and consumption of logic elements.
3. The developed original *FPGA*-based parametrical orthogonal filters allow to perform the extraction of signals of any form better than, for example, wavelet filters.

4. The developed prototype, based on original algorithm and *FPGA*, for the simulator of generalized orthogonal nonsinusoidal division multiplexing allows to reduce essentially simulation time of modeled systems, and to prototype frequency division systems of a new kind – a promising alternative to standard *OFDM*.

Practical Significance of Research

The practical use is to implement the experimental angle-based devices in real *FPGA* devices. Current trial versions of *FPGA* devices can serve as prototypes for new devices, based on angle-based algorithms. The practical results include a variety of experimental devices – *BF* generators, unique orthogonal filters, data transmission system. The automated system *EGURIT* is of especially practical value. It allows to synthesize in *FPGA* numerous simplifications (modifications) of elementary generalized unitary rotation element, using a rotation matrix specified by a formula. *EGURIT* is essential part of the unitary transform implementation system *UNITIT*, which is currently under development.

Approbation

The main ideas of the research have been presented and discussed at the scientific conferences:

- "The 10th International Conference ELECTRONICS", Kaunas, Lithuania, May 23-25, 2006.
- "RTU 48th international scientific conference", Riga, Latvia, October 11-13, 2007.
- "MIXDES-2007", Ciechocinek, Poland, June 20-22, 2007.
- "ECS'07, the 6th Electronic Circuits and Systems Conference", Bratislava, Slovakia, September 6-7, 2007.
- "Norchip 2007", Aalborg, Denmark, November 19-20, 2007.
- "Norchip 2008", Tallinn, Estonia, November 17-18, 2008.
- "Nordic MATLAB User Conference 2008", Stockholm, Sweden, November 20-21, 2008.
- "Norchip 2009", Trondheim, Norway, November 16-17, 2009.
- *"The 15th International Conference ELECTRONICS", Kaunas, Lithuania, May 23-25, 2011.
- *"Norchip 2011", Lund, Sweden, November 14-15, 2011.

* Accepted for publishing

The main ideas have been published in the papers:

[P1] G. Valters, P. Misans, "Initial Version of FPGA-Based CRAIMOT Basis Functions Generator", Proceedings of the 14th IEEE International Conference Mixed Design of Integrated Circuits and Systems MIXDES 2007, The 14th IEEE International Conference Mixed Design of Integrated Circuits and Systems MIXDES 2007, Poland, Ciechocinek, pp. 632-637, June 21.-23, 2007.

[P2] P. Misans, G. Valters, "Initial Version of FPGA-Based CRAIMOT Spectrum Analyzer", Proceedings of the 6th Electronic Circuits and Systems Conference ECS'07, The 6th Electronic Circuits and Systems Conference ECS'07, Slovakia, Bratislava, pp. 159-164, September 6.-7, 2007.

[P3] P. Misans, G. Valters, "Introduction Into The Parametrical Decomposition - Reconstruction Filters Based On Haar-Like Orthonormal Transforms", Proceedings of the 6th International Electronic Circuits and Systems Conference ECS'07, The 6th International Electronic Circuits and Systems Conference ECS'07, Slovakia, Bratislava, pp. 107-112, September 6. -7, 2007.

[P4] P. Misans, M. Terauds, G. Valters, U. Derums, N. Vasilevskis, "FPGA-Based CRAIMOT Basis Function Generator", Proceedings of the 25th IEEE Norchip Conference on CD, The 25th IEEE Norchip Conference, Denmark, Alborg, pp. 1-6, November 19.-20, 2007.

[P5] P. Misans, G. Valters, M. Terauds, A. Aboltins, "Initial Implementation of Generalized Haar-Like Orthonormal Transforms into FPGA-Based Devices - Part I: Signal Spectrum Analyzer-Synthesizer Module", Scientific Journal of RTU. 7. series., Telekomunikācijas un elektronika. Vol. 8, pp 16-21, 2008.

[P6] P. Misans, G. Valters, M. Terauds, N. Vasilevskis, "Initial Implementation of Generalized Haar-Like Orthonormal Transforms into FPGA-Based Devices - Part II: Parametrical Decomposition-Reconstruction Filters", Scientific Journal of RTU. 7. series., Telekomunikācijas un elektronika. Vol. 8, pp 12-16, 2008.

[P7] M. Terauds, G. Valters, U. Derums, N. Vasilevskis, P. Misans, "Comparison of Fixed-Point Arithmetic Errors for the FPGA-Based CRAIMOT Basis Function Generators", Proceedings (on flash memory) of the 26th IEEE Norchip 2008 Conference, ISBN 1-4244-2493-1/08 2008 IEEE, The 26th IEEE Norchip 2008 Conference, Estonia, Tallin, pp. 1-6, November 16.-18, 2008.

[P8] P. Misans, M. Terauds, A. Aboltins, G. Valters, "MATLAB/SIMULINK Implementation of Phi-Transforms – A New Toolbox Only or the Rival of Wavelet Toolbox for the Next Decade?", Proceedings (on CD) of Nordic MATLAB User Conference 2008, Nordic MATLAB User Conference 2008, Sweden, Stockholm, pp. 1-8, November 20.-21, 2008.

[P9] G. Valters, P. Misans, "FPGA Implementation of Elementary Generalized Unitary Rotation", Proceedings of 27th IEEE Norchip 2009 Conference, on the flash, Norway, Trondheim, pp 1-4, November 16.-17, 2009.

[P10] P. Misans, G. Valters, "Initial FPGA Design for Generalized Orthogonal Non-sinusoidal Division Multiplexing", Proceedings of 27th IEEE Norchip 2009 Conference, on the flash, Norway, Trondheim, pp. 1-5, November 16.-17, 2009.

[P11] G. Valters, P. Misans, "Automation of FPGA Implementation of Unitary Transforms Based on Elementary Generalized Unitary Rotation", RTU 52th Scientific Conference 2011, Latvia, Riga, in press.

[P12] G. Valters, "Initial Version of MatLab/Simulink Based Tool for VHDL Code Generation and FPGA Implementation of Elementary Generalized Unitary Rotation", Norchip 2011, Sweden, Lund, in press.

All published papers related to the doctoral thesis:

1. *G. Valters, P. Misans, "Initial Version of FPGA-Based CRAIMOT Basis Functions Generator", Proceedings of the 14th IEEE International Conference Mixed Design of Integrated Circuits and Systems MIXDES 2007, The 14th IEEE International Conference Mixed Design of Integrated Circuits and Systems MIXDES 2007, Poland, Ciechocinek, pp. 632-637, June 21.-23, 2007.

2. P. Misans, G. Valters, "Initial Version of FPGA-Based CRAIMOT Spectrum Analyzer", Proceedings of the 6th Electronic Circuits and Systems Conference ECS'07, The 6th Electronic Circuits and Systems Conference ECS'07, Slovakia, Bratislava, pp. 159-164, September 6.-7, 2007.

3. P. Misans, G. Valters, "Introduction Into The Parametrical Decomposition - Reconstruction Filters Based On Haar-Like Orthonormal Transforms", Proceedings of the 6th International Electronic Circuits and Systems Conference ECS'07, The 6th Inter-

national Electronic Circuits and Systems Conference ECS'07, Slovakia, Bratislava, pp. 107-112, September 6. -7, 2007.

4. *P. Misans, M. Terauds, G. Valters, U. Derums, N. Vasilevskis, "FPGA-Based CRAIMOT Basis Function Generator", Proceedings of the 25th IEEE Norchip Conference on CD, The 25th IEEE Norchip Conference, Denmark, Alborg, pp. 1-6, November 19.-20, 2007.

5. P. Misans, G. Valters, M. Terauds, A. Aboltins, "Initial Implementation of Generalized Haar-Like Orthonormal Transforms into FPGA-Based Devices - Part I: Signal Spectrum Analyzer-Synthesizer Module", Scientific Journal of RTU. 7. series., Telekomunikācijas un elektronika. Vol. 8, pp 16-21, 2008.

6. P. Misans, G. Valters, M. Terauds, N. Vasilevskis, "Initial Implementation of Generalized Haar-Like Orthonormal Transforms into FPGA-Based Devices - Part II: Parametrical Decomposition-Reconstruction Filters", Scientific Journal of RTU. 7. series., Telekomunikācijas un elektronika. Vol. 8, pp 12-16, 2008.

7. *M. Terauds, G. Valters, U. Derums, N. Vasilevskis, P. Misans, "Comparison of Fixed-Point Arithmetic Errors for the FPGA-Based CRAIMOT Basis Function Generators", Proceedings (on flash memory) of the 26th IEEE Norchip 2008 Conference, ISBN 1-4244-2493-1/08 2008 IEEE, The 26th IEEE Norchip 2008 Conference, Estonia, Tallin, pp. 1-6, November 16.-18, 2008.

8. P. Misans, M. Terauds, A. Aboltins, G. Valters, "MATLAB/SIMULINK Implementation of Phi-Transforms – A New Toolbox Only or the Rival of Wavelet Toolbox for the Next Decade?", Proceedings (on CD) of Nordic MATLAB User Conference 2008, Nordic MATLAB User Conference 2008, Sweden, Stockholm, pp. 1-8, November 20.-21, 2008.

9. *G. Valters, P. Misans, "FPGA Implementation of Elementary Generalized Unitary Rotation", Proceedings of 27th IEEE Norchip 2009 Conference, on the flash, Norway, Trondheim, pp 1-4, November 16.-17, 2009.

10. *P. Misans, G. Valters, "Initial FPGA Design for Generalized Orthogonal Nonsinusoidal Division Multiplexing", Proceedings of 27th IEEE Norchip 2009 Conference, on the flash, Norway, Trondheim, pp. 1-5, November 16.-17, 2009.

11. G. Valters, U. Derums, K. Osmanis, P. Misans, "Experimental Image Analyzer-Synthesizer Based on the Novel Discrete Orthogonal Transforms", Electronics 2011, Lithuania, Kaunas, in press.

12. G. Valters, P. Misans, "Automation of FPGA Implementation of Unitary Transforms Based on Elementary Generalized Unitary Rotation", RTU 52th Scientific Conference 2011, Latvia, Riga, in press.

13. *G. Valters, “Initial Version of MatLab/Simulink Based Tool for VHDL Code Generation and FPGA Implementation of Elementary Generalized Unitary Rotation“, Norchip 2011, Sweden, Lund, in press.

* IEEE papers.

Thesis Contents

The review consists of four chapters, bibliography and conclusion. The review is written in Latvian and then translated also into English. It consists of 96 pages, 55 figures, 20 tables and 56 references to sources. The synopsis takes up 19 pages, the description part – 76 pages. The total number of pages in the publications amounts to 70.

Introduction

Fast orthogonal transforms maintain a significant position in a wide range of signal processing algorithms. Over the past decade particularly rapid progress has taken place in the area of wavelets (one of the subclasses of orthogonal functions). For some years now, the research on real 1-D discrete orthogonal transforms is being conducted at the Faculty of Electronics and Telecommunications of Riga Technical University. The research on complex 1-D orthogonal transforms (reviewed in this paper) and real 2-D discrete orthogonal transforms has been started.

Since one of the main parts of the doctoral thesis is author's published scientific papers, a short description of each paper is given below. Detailed information on orthogonal transforms can be found in this review document – Chapter 1 presents theoretical background for rotation-angle-based complex orthogonal transforms and shows their diversity; Chapter 2 deals with various implementation architectures for orthogonal transforms; Chapter 3 reviews problems and their solutions with respect to the transform implementation in *FPGA*; (the last) Chapter 4 reviews experimental rotation-angle-based devices.

Angle-based orthogonal transforms allow to obtain an indefinitely large number, but not all possible, of basis functions. While working on master thesis [14], various generators of real *BFs* have been developed [P1]. At the same time the real-time audio signal spectrum analyzer has been developed [P2]. A Haar-like signal decomposition/reconstruction system based on parametrical orthogonal filters is described in [P3]. [P4] deals with resolving the difficulties with implementation of *BF* generator in *FPGA*. [P5] and [P6] have been published by RTU in the annual collection of scientific proceedings. These two papers describe the implementation of parametrical real Haar-like transforms in *FPGA*. Signal filtering in [P6] is performed using the decomposition/reconstruction system. Errors are inevitable when systems are implemented in real devices with fixed-point arithmetic. [P7] examines *FPA* errors that arise when implementing various algorithms for *BF* generator. For a novel kind of orthogonal transforms a *MatLab* toolbox has been created [P8]. [P9] describes the implementation of a simplified version (see Section 3.4) of complex rotation matrix (see Chapter 1). [P10] describes the trial implementation (a simplified version of complex rotation matrix has been used) of digital part of transmission system based on nonsinusoidal division multiplexing. [P11] discusses the necessity to develop tools for automatic code generation for FPGA implementation of complex rotation matrices. [P12] describes one of such tools and its operation principles.

1 Complex Parametrical Orthogonal Transforms

This chapter presents generalized expressions for phi-transforms that are common for all papers [P1]-[P12]. The notation for generalized Jacobi matrix (see Section 1.2.1) introduced in [P11],[P12] allows to present all the variety of possible phi-transforms – real [P1]-[P7] and complex [P8]-[P12].

1.1 Orthogonal, orthonormal and unitary transforms

Discrete orthogonal transforms are used to calculate the spectrum of discrete signal.

$$\mathbf{Y} = k_1 \cdot \mathbf{H} \cdot \mathbf{X}, \quad \mathbf{X} = k_2 \cdot \mathbf{H}^{-1} \cdot \mathbf{Y} \tag{1.1}$$

where H - Orthogonal transform matrix, $(\dots)^{-1}$ - inverse matrix (for complex transforms, Hermitian matrix). The values of constants k_1 and k_2 depend on specific transform.

Orthonormal transforms and their corresponding matrices are a special case of orthogonal transforms, for which the norm of basis function (matrix row) equals 1. In such a case constants $k_1 = k_2 = 1$ in Formula (1.1). Complex orthonormal matrices are referred to as unitary matrices, and their corresponding transforms as unitary transforms [4]. By calculating the scalar product of matrix rows, it can be checked whether a certain matrix is orthonormal. For orthonormal matrices the following equivalences are valid:

$$\left(\overline{H}_p, H_k\right) = \delta_{p,k}, \qquad \sum_{t=1}^{N} \overline{H}_{p,t} \cdot H_{k,t} = \delta_{p,k}, \tag{1.2}$$

where $\delta_{p,k}$ – Kronecker symbol, H_m – m-th row in matrix $\mathbf{H}$,
$H_{m,q}$ – q-th element of m-th row in matrix $\mathbf{H}$.

1.2 Rotation-angle-based unitary transforms

This section deals with basic elements of complex angle-based orthonormal, that is, unitary, transforms and gives a brief insight into their classification. For more detailed information on the classification of real angle-based orthogonal transforms see [3] and [14].

1.2.1 Elementary generalized unitary rotation matrix

The basic element of novel parametrical unitary transforms is the elementary generalized unitary rotation matrix (further referred to as elementary rotation matrix), which is a generalization of well-known Jacobi rotation matrix [4]. In the case of complex signals the elementary rotation matrix is a three-angle (ϕ, ψ, γ) matrix. When angles $\phi, \psi, \gamma \in [0, 45°]$, 64 different structures of rotation matrix are possible. The number of structures of rotation matrix decreases twice (to 32) when angles $\phi, \psi, \gamma \in [0, 90°]$. Some of the possible

structures of rotation matrix are shown in Table 1. For a convenient way to present all structures of rotation matrix, the notation for generalized Jacobi matrix is introduced [P12]:

$$\mathbf{T}_U = \begin{bmatrix} \begin{pmatrix} - & - \\ - & - \\ + & + \\ + & + \end{pmatrix} \begin{Bmatrix} \sin \\ \cos \end{Bmatrix} e^{\begin{pmatrix} - & + \\ - & + \end{pmatrix} j\psi} & \begin{pmatrix} - & - \\ + & + \\ - & - \\ + & + \end{pmatrix} \begin{Bmatrix} \cos \\ \sin \end{Bmatrix} e^{\begin{pmatrix} - & + \\ + & - \end{pmatrix} j\gamma} \\ \begin{pmatrix} - & + \\ - & + \\ - & + \\ - & + \end{pmatrix} \begin{Bmatrix} \cos \\ \sin \end{Bmatrix} e^{\begin{pmatrix} + & - \\ - & + \end{pmatrix} j\gamma} & \begin{pmatrix} + & - \\ - & + \\ - & + \\ + & - \end{pmatrix} \begin{Bmatrix} \sin \\ \cos \end{Bmatrix} e^{\begin{pmatrix} + & - \\ + & - \end{pmatrix} j\psi} \end{bmatrix}, \tag{1.3}$$

By introducing the notations (1.4) , the matrix (1.3) can be rewritten as (1.5) [P11].

$$\begin{aligned} s_{11} &= \begin{bmatrix} - & - \\ - & - \\ + & + \\ + & + \end{bmatrix}, \quad s_{12} = \begin{bmatrix} - & - \\ + & + \\ - & - \\ + & + \end{bmatrix}, \quad s_{21} = \begin{bmatrix} - & + \\ - & + \\ - & + \\ - & + \end{bmatrix}, \quad s_{22} = \begin{bmatrix} + & - \\ - & + \\ - & + \\ + & - \end{bmatrix}, \\ sc &= \begin{bmatrix} \sin(\phi) \\ \cos(\phi) \end{bmatrix}, \quad cs = \begin{bmatrix} \cos(\phi) \\ \sin(\phi) \end{bmatrix}, \quad sp = \begin{bmatrix} - & + \end{bmatrix}, \quad sg = \begin{bmatrix} + & - \\ - & + \end{bmatrix} \end{aligned} \tag{1.4}$$

$$\mathbf{T}(\phi, \psi, \gamma, a, b, u) = \begin{bmatrix} s_{11}(a)sc(b) \cdot e^{sp(u)j\cdot\psi} & s_{12}(a)cs(b) \cdot e^{-sg(u)j\cdot\gamma} \\ s_{21}(a)cs(b) \cdot e^{sg(u)j\cdot\psi} & s_{22}(a)sc(b) \cdot e^{-sp(u)j\cdot\psi} \end{bmatrix} \tag{1.5}$$

So, for example, the following structure is obtained when choosing actual values for a, b, u – 4th element in sign matrix, 2nd element in *sin/cos* matrix, 4th element in sign matrix for the power of exponent:

$$\mathbf{T}_U\left[\phi, \psi, \gamma, 4, 2, 4\right] = \begin{bmatrix} \cos(\phi) \cdot e^{i\cdot\psi} & \sin(\phi) \cdot e^{-i\cdot\gamma} \\ -\sin(\phi) \cdot e^{i\cdot\gamma} & \cos(\phi) \cdot e^{-i\cdot\psi} \end{bmatrix} \tag{1.6}$$

For the sake of simplicity the parameters a, b, u can be presented as a single index $id3$ [P11]:

$$\mathbf{T}_U\left(\phi, \psi, \gamma, 424\right) = \begin{bmatrix} \cos(\phi) \cdot e^{i\cdot\psi} & \sin(\phi) \cdot e^{-i\cdot\gamma} \\ -\sin(\phi) \cdot e^{i\cdot\gamma} & \cos(\phi) \cdot e^{-i\cdot\psi} \end{bmatrix} \tag{1.7}$$

The structure parameters can be simply obtained from $id3$ using the following formulas:

$$\begin{aligned} &a = fix(id3/100),\ b = round(fix(id3/10) - 10\times a), \\ &u = round(10\times(id3/10 - fix(id3/10))), \end{aligned} \tag{1.8}$$

where fix and $round$ – two types of rounding a number (more detailed described in Section 3.2.1).

It is sometimes rather difficult to use the index $id3$ in automated algorithms, because, for example, after 114 there follows 121. For that reason the index $id1 \in [1, 64], id1 \in \mathbb{N}$ is introduced and it can be calculated as follows:

$$id1 = (a-1)\cdot 8 + (b-1)\cdot 4 + u \tag{1.9}$$

Table 1 shows some of the possible 64 structures of Jacobi rotation matrix and all the

Table 1: Some random examples of rotation matrix shapes for different index sets [P11]

ID *id1*	*Index set* $\{a,\ b,\ u\}$, *id3*	*Rotation matrix*
1	$\{1,\ 1,\ 1\}$ 111	$\begin{bmatrix} -\sin(\phi)\cdot e^{-j\psi} & -\cos(\phi)\cdot e^{-j\gamma} \\ -\cos(\phi)\cdot e^{+j\gamma} & +\sin(\phi)\cdot e^{+j\psi} \end{bmatrix}$
2	$\{1,\ 1,\ 2\}$ 112	$\begin{bmatrix} -\sin(\phi)\cdot e^{-j\psi} & +\cos(\phi)\cdot e^{-j\gamma} \\ -\cos(\phi)\cdot e^{+j\gamma} & -\sin(\phi)\cdot e^{+j\psi} \end{bmatrix}$
3	$\{1,\ 1,\ 3\}$ 113	$\begin{bmatrix} +\sin(\phi)\cdot e^{-j\psi} & -\cos(\phi)\cdot e^{-j\gamma} \\ -\cos(\phi)\cdot e^{+j\gamma} & -\sin(\phi)\cdot e^{+j\psi} \end{bmatrix}$
62	$\{8,\ 2,\ 2\}$ 822	$\begin{bmatrix} +\cos(\phi)\cdot e^{-j\psi} & +\sin(\phi)\cdot e^{+j\gamma} \\ +\sin(\phi)\cdot e^{-j\gamma} & -\cos(\phi)\cdot e^{+j\psi} \end{bmatrix}$
63	$\{8,\ 2,\ 3\}$ 823	$\begin{bmatrix} +\cos(\phi)\cdot e^{+j\psi} & +\sin(\phi)\cdot e^{+j\gamma} \\ +\sin(\phi)\cdot e^{-j\gamma} & -\cos(\phi)\cdot e^{-j\psi} \end{bmatrix}$
64	$\{8,\ 2,\ 4\}$ 824	$\begin{bmatrix} +\cos(\phi)\cdot e^{+j\psi} & +\sin(\phi)\cdot e^{-j\gamma} \\ +\sin(\phi)\cdot e^{+j\gamma} & -\cos(\phi)\cdot e^{-j\psi} \end{bmatrix}$

described ways of indexing used for the generalized matrix. Real rotation matrices, for example, the Givens rotation matrix, are used in [3] and [14]. Note, that real rotation matrix may be considered as a simplified case of complex matrix (1.10). [P1]-[P7] deal with real orthogonal transforms and corresponding devices.

$$\mathbf{T}_U(\phi,\ \psi=0,\ \gamma=0,\ 424) = \begin{bmatrix} \cos(\phi) & \sin(\phi) \\ -\sin(\phi) & \cos(\phi) \end{bmatrix} \tag{1.10}$$

A complex vector obtained by rotating a complex input vector $\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$ by angles ϕ, ψ, γ, can be considered as two-sample signal spectrum $\begin{bmatrix} y_1 \\ y_2 \end{bmatrix}$ dependent on angles ϕ, ψ and γ:

$$\mathbf{Y} = \mathbf{T}_U \cdot \mathbf{X} = \begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = \mathbf{T}_U \cdot \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \tag{1.11}$$

Two FIR filters with the coefficients $b_L = [\cos(\phi)\cdot e^{i\cdot\gamma},\ \sin(\phi)\cdot e^{-i\cdot\psi}]$ and $b_H = [-\sin(\phi)\cdot e^{i\cdot\psi},\ \cos(\phi)\cdot e^{-i\cdot\gamma}]$ are chosen and downsampling of filter outputs performed. After processing of two input samples the same signal spectrum is obtained (see Figure 1.1).

Orthogonal filters and their properties are described more detailed in Section 2.4. To reconstruct a signal, the obtained spectrum must be multiplied by inverse $\mathbf{T}^{-1}$ matrix (in the case of complex matrix, by Hermitian matrix) [P9]:

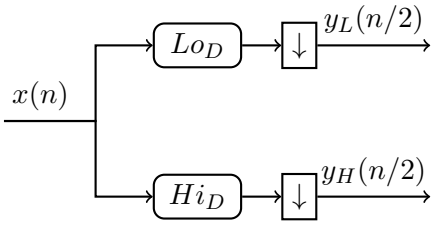


Figure 1.1: Decomposition of two signal samples

$$\mathbf{T}_U^{-1}(\phi,\ \psi = 0,\ \gamma = \frac{\pi}{2},\ 424) = \begin{bmatrix} \cos(\phi) \cdot e^{-i\gamma} & -\sin(\phi) \cdot e^{-i\psi} \\ \sin(\phi) \cdot e^{i\psi} & \cos(\phi) \cdot e^{i\gamma} \end{bmatrix} \tag{1.12}$$

Geometrically signal reconstruction means the rotation of complex vector in the opposite direction.

$$\mathbf{X}_{rec} = \mathbf{T}_U^{-1} \cdot \mathbf{Y} \tag{1.13}$$

1.2.2 Full unitary matrix

The length of used discrete signals is usually more than two samples. Further in this paper signals whose length is $N = 2^n$ samples are considered. In N-dimensional space a simultaneous rotation of N-dimensional vector can be easily performed in $N/2$ independent planes. There are several ways to construct a matrix to perform mentioned rotations. One of the ways is to use a stairs-like factorized matrix. Using the structure $\mathbf{T}_U(\phi,\ \psi,\ \gamma,\ 424)$, the following 4×4 unitary stairs-like matrix is obtained:

$$\mathbf{B}(\phi,\ \psi,\ \gamma) = \begin{bmatrix} ce^{i\psi} & se^{-i\gamma} & 0 & 0 \\ 0 & 0 & ce^{i\psi} & se^{-i\gamma} \\ -se^{i\gamma} & ce^{-i\psi} & 0 & 0 \\ 0 & 0 & -se^{i\gamma} & ce^{-i\psi} \end{bmatrix} \tag{1.14}$$

where $c = \cos(\phi),\ s = \sin(\phi)$

For matrix presentation it is much more convenient to use blocks. By presenting a matrix in such a way, independent rotation structures are easily recognized.

$$\mathbf{B}_b(\phi, \psi, \gamma) = \begin{bmatrix} ce^{i\psi} & se^{-i\gamma} & 0 & 0 \\ -se^{i\gamma} & ce^{-i\psi} & 0 & 0 \\ 0 & 0 & ce^{i\psi} & se^{-i\gamma} \\ 0 & 0 & -se^{i\gamma} & ce^{-i\psi} \end{bmatrix} \tag{1.15}$$

However, a stairs-like matrix has an essential advantage in comparison to a block-like matrix – when multiplying $\log_2(N)$ stairs-like matrices, a "full" matrix, in common case containing no zeros, is obtained. For example, a 4×4 Hadamard matrix (which is, by

the way, orthogonal, instead of orthonormal) can be obtained using two equal stairs-like matrices (from Hadamard fast transform [16]):

$$\mathbf{H}_{Had4} = \sqrt{2}\cdot\begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0 & 0 \\ 0 & 0 & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} & 0 & 0 \\ 0 & 0 & \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \end{bmatrix}\cdot\sqrt{2}\cdot\begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0 & 0 \\ 0 & 0 & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} & 0 & 0 \\ 0 & 0 & \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{bmatrix} \tag{1.16}$$

A stairs-like matrix can be obtained from a block-like matrix (and vice versa), by rearranging matrix rows – using an appropriate permutation matrix:

$$\mathbf{B}_s(\phi,\ \psi,\ \gamma) = \mathbf{P}_4 \cdot \mathbf{B}_b(\phi,\ \psi,\ \gamma), \tag{1.17}$$

where

$$\mathbf{P}_4 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{1.18}$$

Similarly to (1.16), from two stairs-like matrices a "full" 4×4 unitary transform matrix is obtained (1.19). For each rotation it is possible to use a specific set of angles. For convenience the following notation is introduced – $\Phi_k = (\phi_k, \psi_k, \gamma_k)$.

$$\mathbf{H}_4 = \mathbf{B}_s\left(\Phi_{11}, \Phi_{12}\right)\cdot\mathbf{B}_s\left(\Phi_{21}, \Phi_{22}\right) =$$
$$= \begin{pmatrix} c_{11}\,c_{21}\,e^{\psi_{11}\,i}\,e^{\psi_{21}\,i} & c_{11}\,s_{21}\,e^{-\gamma_{21}\,i}\,e^{\psi_{11}\,i} & c_{22}\,s_{11}\,e^{-\gamma_{11}\,i}\,e^{\psi_{22}\,i} & s_{11}\,s_{22}\,e^{-\gamma_{11}\,i}\,e^{-\gamma_{22}\,i} \\ -c_{12}\,s_{21}\,e^{\gamma_{21}\,i}\,e^{\psi_{12}\,i} & c_{12}\,c_{21}\,e^{\psi_{12}\,i}\,e^{-\psi_{21}\,i} & -s_{12}\,s_{22}\,e^{-\gamma_{12}\,i}\,e^{\gamma_{22}\,i} & c_{22}\,s_{12}\,e^{-\gamma_{12}\,i}\,e^{-\psi_{22}\,i} \\ -c_{21}\,s_{11}\,e^{\gamma_{11}\,i}\,e^{\psi_{21}\,i} & -s_{11}\,s_{21}\,e^{\gamma_{11}\,i}\,e^{-\gamma_{21}\,i} & c_{11}\,c_{22}\,e^{-\psi_{11}\,i}\,e^{\psi_{22}\,i} & c_{11}\,s_{22}\,e^{-\gamma_{22}\,i}\,e^{-\psi_{11}\,i} \\ s_{12}\,s_{21}\,e^{\gamma_{12}\,i}\,e^{\gamma_{21}\,i} & -c_{21}\,s_{12}\,e^{\gamma_{12}\,i}\,e^{-\psi_{21}\,i} & -c_{12}\,s_{22}\,e^{\gamma_{22}\,i}\,e^{-\psi_{12}\,i} & c_{12}\,c_{22}\,e^{-\psi_{12}\,i}\,e^{-\psi_{22}\,i} \end{pmatrix} \tag{1.19}$$

where

$$\mathbf{B}_s\left(\Phi_{11}, \Phi_{12}\right) = \begin{bmatrix} c_{12}e^{i\psi_{12}} & s_{12}e^{-i\gamma_{12}} & 0 & 0 \\ 0 & 0 & c_{21}e^{i\psi_{21}} & s_{21}e^{-i\gamma_{21}} \\ -s_{12}e^{i\gamma_{12}} & c_{12}e^{-i\psi_{12}} & 0 & 0 \\ 0 & 0 & -s_{21}e^{i\gamma_{21}} & c_{21}e^{-i\psi_{21}} \end{bmatrix} \tag{1.20}$$

The obtained matrix $\mathbf{H}_4$ is a unitary matrix for any values of parameters. It can be verified by performing the scalar multiplication of complex matrix rows:

$$\overline{H}_{n,*} \cdot H_{n,*} = \sum_{k=1}^{N} \overline{H}_{n,k} H_{n,k} = 1,$$
$$\overline{H}_{n,*} \cdot H_{m,*} = \sum_{k=1}^{N} \overline{H}_{m,k} H_{m,k} = 0 \tag{1.21}$$

where H_n – n-th row in matrix H.

Similarly complex orthonormal matrices of larger size can be obtained. The matrix size must be:

$$N = 2^n, \quad n \in \{1, 2, ...k\}. \tag{1.22}$$

The analytic notation (1.19) is too sophisticated even for $N = 4$. It is given here to show how the generalized complex rotation-angle-based orthogonal transform (**CRABOT**) matrix is obtained. The necessary number of parameters is $n_{par} = 3 \cdot \frac{N}{2} \cdot \log_2(N)$. Assemble all possible parameters (that is, angles) into a single three-layered matrix (1.23) [P1], [P5], [P8], [P10], [P11].

$$\mathbf{\Phi} = \begin{bmatrix} \Phi_{11} & \Phi_{12} & \Phi_{13} & \dots & \Phi_{1n} \\ \Phi_{21} & \Phi_{22} & \Phi_{23} & \dots & \Phi_{2n} \\ \Phi_{31} & \Phi_{32} & \Phi_{33} & \dots & \Phi_{3n} \\ \dots & \dots & \dots & \dots & \dots \\ \Phi_{\frac{N}{2}1} & \Phi_{\frac{N}{2}2} & \Phi_{\frac{N}{2}3} & \dots & \Phi_{\frac{N}{2}n} \end{bmatrix} \tag{1.23}$$

Denote with $\mathbf{\Phi}_p$ the p-th column of matrix $\mathbf{\Phi}$. If

$$\mathbf{\Phi}_p = \begin{bmatrix} \Phi_{1p}|_{=\Phi_p} \\ \Phi_{2p}|_{=\Phi_p} \\ \dots \\ \Phi_{\frac{N}{2}p}|_{=\Phi_p} \end{bmatrix} = (\phi_p, \psi_p, \gamma_p), \tag{1.24}$$

where $\mathbf{\Phi}_p$ – p-th column in the angle matrix, Φ_p – set of angles,

then one of the simplified angle-based transforms is obtained – **CCRAIMOT** (***C**omplex **C**onstant **A**ngle **I**n **M**atrix **O**rthogonal **T**ransform*). Using this transform class, an indefinitely large number, but not all possible, of complex orthogonal matrices can be obtained, and it can be controlled using $n_{Par} = 3 \cdot \log_2 N$ parameters (angles).

An example of **CCRAIMOT** matrix for $N = 8$:

$$\mathbf{H}_8 = \mathbf{B}_s(\Phi_3) \cdot \mathbf{B}_s(\Phi_2) \cdot \mathbf{B}_s(\Phi_1) \quad \begin{cases} \Phi_1 = (60^\circ, 60^\circ, 60^\circ) \\ \Phi_2 = (60^\circ, -60^\circ, 60^\circ) \\ \Phi_3 = (60^\circ, -60^\circ, -60^\circ) \end{cases} \tag{1.25}$$

Figure 1.2 shows *BFs* for (1.25).

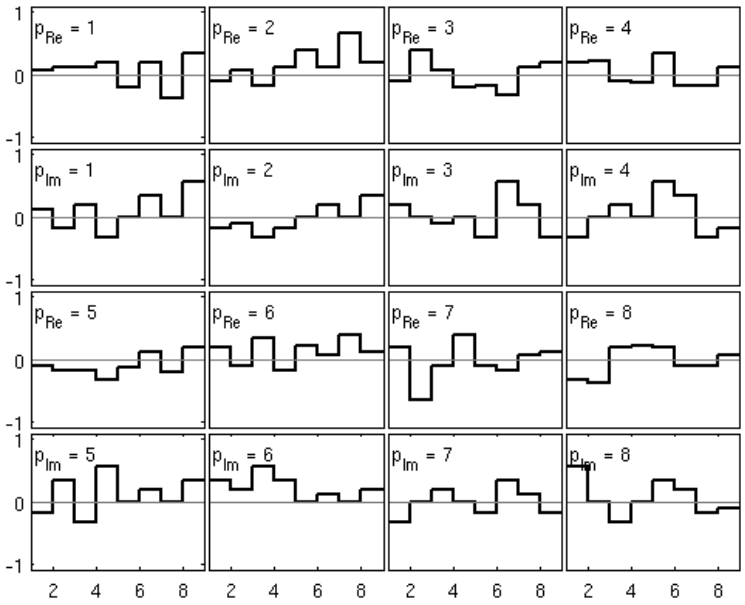


Figure 1.2: **CCRAIMOT** basis functions, $N = 8$,
$\phi = [60°, 60°, 60°]$, $\psi = [60°, -60°, -60°]$, $\gamma = [60°, 60°, -60°]$

If $N = 2^n, n > 2, n \in \mathbb{N}$, then (1.25) may be generalized as:

$$\mathbf{H} = \prod_{p=l}^{1} \mathbf{B}_s \left(\Phi_p\right), \quad l = \log_2 N \tag{1.26}$$

The decomposition of N samples long signal in matrix form can be performed using the fast algorithm [17], [18] – multiplying the input signal by $\log_2(N)$ stairs-like matrices (1.27).

$$\mathbf{Y} = \mathbf{B}_s \left(\Phi_l\right) \cdot \mathbf{B}_s \left(\Phi_{l-1}\right) \cdot_{\ldots} \cdot \mathbf{B}_s \left(\Phi_1\right) \cdot \mathbf{X}, \quad l = \log_2 N \tag{1.27}$$

Table 2 shows a comparison of fast and ordinary algorithms. The number of multiplications (M) and the number of additions (A) are given. The number of multiplications and additions, necessary to multiply an $N \times N$ complex matrix by an N samples long complex vector, is calculated using the following formulas:

$$\begin{aligned} M_C &= 4 \cdot N^2 \\ A_C &= N \cdot (2 \cdot N + (N-1) \cdot 2) \end{aligned} \tag{1.28}$$

Using the fast algorithm, the number of arithmetic operations is the following:

$$\begin{aligned} M_F &= 2^3 \cdot N \cdot \log_2(N) \\ A_F &= 6 \cdot N \cdot \log_2(N) \end{aligned} \tag{1.29}$$

Table 2: Comparison of fast and ordinary algorithms. The number of arithmetic operations

N	**Fast**		**Ordinary**	
	M	A	M	A
2	16	12	16	12
4	64	48	64	56
8	192	144	256	240
16	512	384	1024	992
32	1280	960	4096	4032
64	3072	2304	16384	16256

1.2.3 Haar-like complex orthogonal transforms

The *MatLab* Toolbox presented in [P8] (see Table 2 in [P8]) includes parametrical Haar-like complex orthogonal transforms reviewed in this section. The construction of these transforms is similar to the construction of real generalized, that is, parametrical, Haar-like transforms, which is thoroughly described in [15]. Due to the restricted size of published papers, [P8] and [15] do not present more detailed expressions for complex transform, therefore a brief review is given here.

Using elementary complex rotation matrices, complex parametrical Haar transforms can be obtained. In the special case when $\phi = \pi/4$, $\psi = 0$, $\gamma = 0$, an elementary complex rotation matrix is a usual elementary Haar matrix:

$$\mathbf{T}_U(814) = \begin{bmatrix} se^{i\psi} & ce^{-i\gamma} \\ ce^{i\gamma} & -se^{-i\psi} \end{bmatrix}, \quad \mathbf{T}_U(\phi = \frac{\pi}{4}, \psi = 0, \gamma = 0,\ 814) = \Phi_2 = \frac{\sqrt{2}}{2} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \tag{1.30}$$

A Haar matrix of larger size differs from the matrices described in Section 1.2.2 – all Haar functions (matrix rows), except first two, have a localized energy, that is, adjacent matrix elements have nonzero values (in a local interval). The parametrical Haar matrix is obtained from $\log_2(N)$ factorized matrices, like in Section 1.2.2. For Haar transforms, unlike for those in previous section, when the matrix size changes, the structure and the number of parameters of factorized matrices also change. For example, the generalized Haar-like matrix for $N = 4$ is obtained as follows:

$$\mathbf{H}_H(\Phi_{11},\Phi_{21},\Phi_{12}) =$$
$$= \begin{bmatrix} s_{21}e^{i\psi_{21}} & c_{21}e^{i\gamma_{21}} & 0 & 0 \\ c_{21}e^{-i\gamma_{21}} & -s_{21}e^{-i\psi_{21}} & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} s_{11}e^{i\psi_{11}} & c_{11}e^{i\gamma_{11}} & 0 & 0 \\ 0 & 0 & s_{12}e^{i\gamma_{12}} & c_{12}e^{i\psi_{12}} \\ c_{11}e^{-i\psi_{11}} & -s_{11}e^{-i\gamma_{11}} & 0 & 0 \\ 0 & 0 & c_{12}e^{-i\gamma_{12}} & -s_{12}e^{-i\psi_{12}} \end{bmatrix} \tag{1.31}$$

In (1.31) the factorized matrix on the left side is made of an elementary unitary rotation matrix, zeros and a unit matrix. For input parameters ($\phi = 0, \psi = 0, \gamma = 0$), some blocks of complex rotation matrix are a unit matrix. For convenience, further when $\Phi_k = 0$, those blocks of elementary rotation matrix are chosen, for which the following is valid:

$$\mathbf{T}_U\left(\Phi|_{=0}\right) = \mathbf{I}_2 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \tag{1.32}$$

If the structure of factorized matrices is known, the Haar-like transform matrix for $N = 4$ (1.31) can be expressed using the following angle matrix:

$$\mathbf{\Phi}_H = \begin{bmatrix} \Phi_{11} & \Phi_{12} \\ \Phi_{21} & 0 \end{bmatrix} \tag{1.33}$$

In common case, when $N = 2^n, n > 2, n \in \mathbb{N}$ and taking into account the condition (1.32), the Haar-like transform matrix can be presented using the angle matrix [P3], [P5], [P8] so, that the next column of angle matrix has twice less nonzero angles as the previous one.

$$\mathbf{\Phi}_H = \begin{bmatrix} \Phi_{1,1} & \dots & \Phi_{1,n-2} & \Phi_{1,n-1} & \Phi_{1,n} \\ \Phi_{2,1} & \dots & \Phi_{2,n-2} & \Phi_{2,n-1} & 0 \\ \Phi_{3,1} & \dots & \Phi_{3,n-2} & 0 & 0 \\ \Phi_{4,1} & \dots & \Phi_{4,n-2} & 0 & 0 \\ \Phi_{5,1} & \dots & 0 & 0 & 0 \\ \dots & \dots & \dots & \dots & \dots \\ \Phi_{\frac{N}{2}1} & \dots & 0 & 0 & 0 \end{bmatrix} \tag{1.34}$$

Similarly to Section 1.2.2, Haar-like transforms also can be divided into subclasses. For example, for a single angle set Φ_k for each factorized matrix, similarly to **CCRAIMOT** class in Section 1.2.2, the **CRAIM-HT** – Constant Rotation Angle In Matrix - Haar Transform can be obtained. Using the permutation matrix $\mathbf{P}_4$ (1.18), the universal notation of complex matrix $\mathbf{T}_U$, a zero matrix $\mathbf{0}$ and a unit matrix $\mathbf{I}$, then the 4-sample **CRAIM-HT** transform, can be presented as follows:

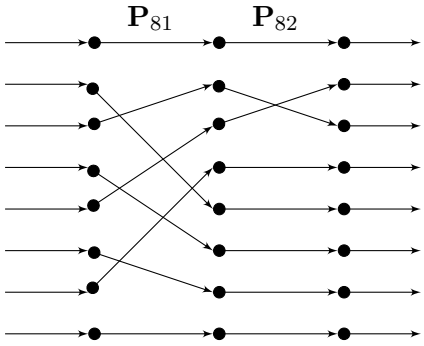


Figure 1.3: Graphical depiction of permutation matrices for Haar-like transforms ($N = 8$)

$$\mathbf{Y} = \mathbf{H}_{H4}(\mathbf{\Phi}_H) \cdot \mathbf{X} = \begin{bmatrix} \mathbf{T}_U(\Phi_2) & \mathbf{0}_2 \\ \mathbf{0}_2 & \mathbf{I}_2 \end{bmatrix} \cdot \mathbf{P}_4 \cdot \begin{bmatrix} \mathbf{T}_U(\Phi_1) & \mathbf{0}_2 \\ \mathbf{0}_2 & \mathbf{T}_U(\Phi_1) \end{bmatrix} \cdot \mathbf{X}, \tag{1.35}$$

where $\mathbf{\Phi}_H$ – angle matrix (1.33) for the transform, and an index at $\mathbf{I}$ and $\mathbf{0}$ indicates the size of unit or zero matrix, respectively.

To obtain the Haar-like transform matrix for $N = 8$, the structure of factorized matrices, which depends on permutation matrices (whose formation algorithm can be found in [15]) and the corresponding angle matrix, must be known. The use of permutation matrices facilitates the notation of expressions, because it allows to present factorized matrices using complex rotation matrix $\mathbf{T}_U$. The graphical depiction of permutation matrices

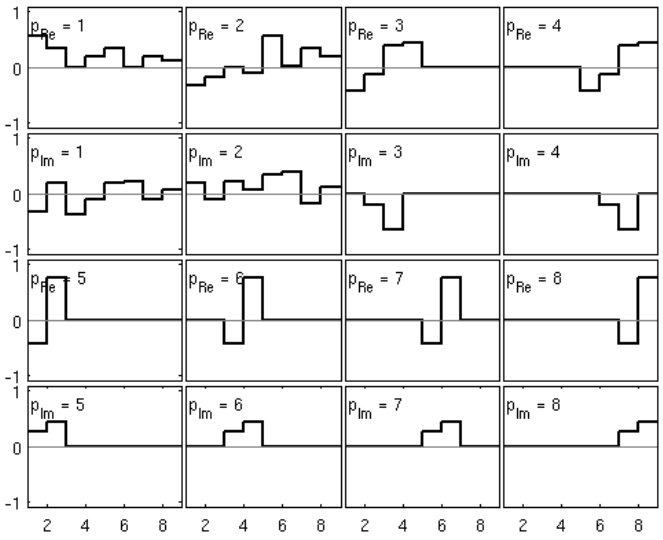


Figure 1.4: **CRA-HT** basis functions, $N = 8$,
$\phi = [30°, 30°, 30°]$, $\psi = [-30°, 30°, -30°]$, $\gamma = [-30°, 30°, -30°]$

(Figure 1.3) helps for understanding their construction, and it is important for obtaining Haar-like transforms for $N = 2^n$, where $n > 2$ and $n \in \mathbb{N}$.

Using permutation matrices from Figure 1.3, the **CRAIM-HT** transform matrix is expressed as follows:

$$\mathbf{H}_{H8}(\Phi_1, \Phi_2, \Phi_3) = \begin{bmatrix} \mathbf{T}_U(\Phi_3) & \mathbf{0}_{2,6} \\ \mathbf{0}_{6,2} & \mathbf{I}_6 \end{bmatrix} \cdot \mathbf{P}_{82} \cdot \begin{bmatrix} \mathbf{T}_U(\Phi_2) & \mathbf{0}_2 & & \mathbf{0}_4 \\ \mathbf{0}_2 & \mathbf{T}_U(\Phi_2) & & \\ & & \mathbf{0}_4 & \\ & & & \mathbf{I}_4 \end{bmatrix} \cdot$$
$$\cdot \mathbf{P}_{81} \cdot \begin{bmatrix} \mathbf{T}_U(\Phi_1) & \mathbf{0}_2 & & \\ \mathbf{0}_2 & \mathbf{T}_U(\Phi_1) & & \\ & & \mathbf{T}_U(\Phi_1) & \mathbf{0}_2 \\ & \mathbf{0}_4 & \mathbf{0}_2 & \mathbf{T}_U(\Phi_1) \end{bmatrix} \tag{1.36}$$

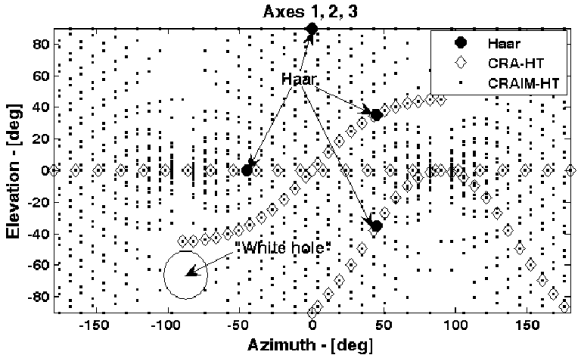
Figure 1.4 shows all **CRAIM-HT basis** functions for the angle matrix:

$$\mathbf{\Phi}_H = \begin{bmatrix} \Phi_1 & \Phi_2 & \Phi_3 \\ \Phi_1 & \Phi_2 & 0 \\ \Phi_1 & 0 & 0 \\ \Phi_1 & 0 & 0 \end{bmatrix} = \begin{bmatrix} (30°, -30°, -30°) & (30°, 30°, 30°) & (30°, -30°, -30°) \\ (30°, -30°, -30°) & (30°, 30°, 30°) & (0, 0, 0) \\ (30°, -30°, -30°) & (0, 0, 0) & (0, 0, 0) \\ (30°, -30°, -30°) & (0, 0, 0) & (0, 0, 0) \end{bmatrix} \tag{1.37}$$

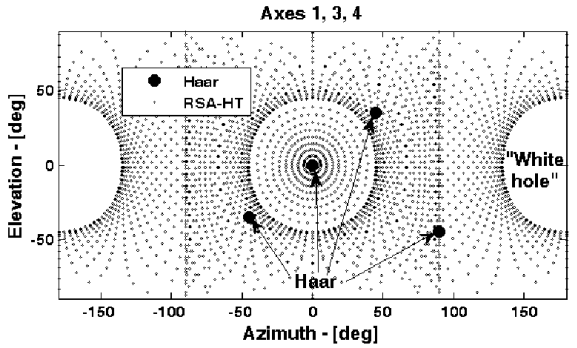
Table 3 shows the division of Haar-like transforms into subclasses depending on the restrictions on nonzero elements in the angle matrix $\mathbf{\Phi}_H$. Formally, the description of complex Haar-like transforms [P8] is similar to the description of real Haar-like transforms

Table 3: Summary of the subclasses of **CRA-HT** transforms [P3], [P5], [P8]

Subclass	Restriction for angles	Example (N=8)
Complex **C**onstant **R**otation **A**ngle **HT** (**CCRA-HT**)	$\mathbf{\Phi}_j = [\overbrace{\Phi,\ \Phi,\ ...,\ \Phi}^{N/(2^j)},\ \underbrace{0,\ ,...,\ 0}_{N/2}]^T$	$\mathbf{\Phi}_H = \begin{bmatrix} \Phi & \Phi & \Phi \\ \Phi & \Phi & 0 \\ \Phi & 0 & 0 \\ \Phi & 0 & 0 \end{bmatrix}$
Complex **C**onstant **R**otation **A**ngle **I**n **M**atrix **HT** (**CCRAIM-HT**)	$\mathbf{\Phi}_j = [\overbrace{\Phi_j,\ \Phi_j,\ ...,\ \Phi_j}^{N/(2^j)},\ \underbrace{0,\ ,...,\ 0}_{N/2}]^T$	$\mathbf{\Phi}_H = \begin{bmatrix} \Phi_1 & \Phi_2 & \Phi_3 \\ \Phi_1 & \Phi_2 & 0 \\ \Phi_1 & 0 & 0 \\ \Phi_1 & 0 & 0 \end{bmatrix}$
Complex **HT** with **R**educed **S**equences of **R**otation **A**ngles (**CRSA-HT**)	$\mathbf{\Phi}_j = [\overbrace{\Phi_1,\ \Phi_2,\ ...,\ \Phi_{f(j)}}^{f(j)=N/(2^j)},\ \underbrace{0,\ ,...,\ 0}_{N/2}]^T$	$\mathbf{\Phi}_H = \begin{bmatrix} \Phi_1 & \Phi_1 & \Phi_1 \\ \Phi_2 & \Phi_2 & 0 \\ \Phi_3 & 0 & 0 \\ \Phi_4 & 0 & 0 \end{bmatrix}$



(a) **CRAIM-HT** un **CRA-HT**



(b) **RSA-HT**

Figure 1.5: Trace of *BFs* vector end projections on the spread of surface of one of the unit sphere 3-D projections [P5]

[P3], [P5].

It is hard to visualize a four-dimensional space (as well as any other multidimensional space when $N > 4$). However, any three samples of four-dimensional *BFs* can be displayed as projections in three-dimensional space. In total there are $C_4^3 = 4$ 3-D projections of 4-D signal possible. By marking vector end projections on a 3-D sphere and then spreading this sphere in a 2-D plane, a certain layout of *BFs* vector end projections is obtained (see Figure 1.5). Fully covered spheres for all projections would signify that for the given transform class, by changing angles, it is possible to obtain *BFs* of any shape. As can be seen, the general tendency is such that by increasing the number of angles (parameters), the coverage becomes more and more dense.

Summary

Obtained results

- Generalized notation for elementary unitary matrix has been introduced. As a result, all possible structures for generalizations of Jacobi matrix are obtained (in case of $\phi, \psi, \gamma \in [0, 45°]$, 64 structures are possible).

Conclusions

- By changing angles ϕ,ψ and γ, an indefinitely large number of various phi-transforms can be obtained.
- Modification of factorized matrices as a result of choosing a part of rotation angles to be equal to zero, allows to obtain pulse-like *BFs* (generalized parametrical Haar-like functions similar to wavelets).
- Using of fast algorithms essentially reduces the number of arithmetic operations for the transforms described in Section 1.2.2 when $N > 4$, and for Haar-like transforms when $N > 8$.

2 Implementation Architectures for Orthogonal Transforms

This chapter presents theoretical aspects of implementation architectures for phi-transforms[6]:

1. Parallel structure of transform implementation,
2. Serial structure of transform implementation,
3. Tree-like structure of signal decomposition/reconstruction system,
4. Signal decomposition using complex *FIR* filters.

2.1 Parallel structure of transform implementation

By directly implementing the expression (1.27), the parallel structure of transform is obtained. The transform is performed by the following steps:

1. Collect a data block of N samples;
2. Multiply the samples by $\log_2(N)$-number of sparse stairs-like rotation matrices $\mathbf{B}_s$;
3. Output the results and simultaneously collect samples for the next data block.

The Analyzer-Synthesizer from [P5] can also be regarded as parallel architecture of transform implementation, because the transform is performed in a similar way as just described. However, unlike previously, the collected pairs of samples are regarded as two-dimensional signal vectors and rotated along several parallel *CORDIC RE* (see Section 3.3).

2.2 Serial structure of transform implementation

A serial structure of transform implementation is a structure when the transform is not performed directly, but instead any value of the phi-transform matrix **H** can be obtained using the below-described algorithm.

2.2.1 CRAIMOT *BF* generator [P1], [P4]

This section examines the generation of real **CRAIMOT** *BFs* and compares the following two generators in respect to the use of arithmetic operations:

1. Generating of the matrix **H** for **CRAIMOT** transform, using the fast algorithm.
2. Generating of *BF* samples, using *sin/cos* products.

[6]The developed experimental devices are described in Chapter 4.

The number of arithmetic operations required for calculation of real transform matrix using the fast algorithm, is determined as follows [P4]:

$$\begin{aligned} M_{FT} &= 2 \cdot N \cdot \log_2(N), \\ A_{FT} &= N \cdot \log_2(N) \end{aligned} \tag{2.1}$$

To store all *BFs*, considerable memory resources (N^2 memory cells) are required. Using the *sin/cos* algorithm, the number of operations for generating a K-number of *BFs* is determined as follows [P4]:

$$\begin{aligned} M_M &= K \cdot N, \\ A_M &= K \cdot (N-1). \end{aligned} \tag{2.2}$$

The serial architecture is based on generating any *BF* using *sin/cos* products – the t-th value of the p-th *BF* is obtained by the following formula [P1], [P4], [3]:

$$\mathbf{H}\,(p-1,\ t-1) = \prod_{j=1}^{\log_2(N)} (-1)^{a_j(p,t)} \cdot s_j^{m_j(p,t)} \cdot c_j^{k_j(p,t)} \tag{2.3}$$

The values of powers (a_j, m_j un k_j) depend on the structure of elementary rotation matrix (see Table 4).

Table 4: Expressions for the powers of sine and cosine (taken from [3], complemented)

Rotation matrix	*ID*	$a_j(p,t)$	$m_j(p,t)$	$k_j(p,t)$
$\mathbf{T}_U(\phi,\ 0,\ 0,\ 421)$	$\mathbf{T}_{G+}$	$\overline{p}_j \cdot t_j$	$\overline{p}_j \cdot t_j + p_j \cdot \overline{t}_j$	$p_j \cdot t_j + \overline{p}_j \cdot \overline{t}_j$
$\mathbf{T}_U(\phi,\ 0,\ 0,\ 721)$	$\mathbf{T}_{G-}$	$p_j \cdot \overline{t}_j$	$\overline{p}_j \cdot t_j + p_j \cdot \overline{t}_j$	$p_j \cdot t_j + \overline{p}_j \cdot \overline{t}_j$
$\mathbf{T}_U(\phi,\ 0,\ 0,\ 611)$	$\mathbf{T}_{R+}$	$p_j \cdot t_j$	$p_j \cdot t_j + \overline{p}_j \cdot \overline{t}_j$	$\overline{p}_j \cdot t_j + p_j \cdot \overline{t}_j$
$\mathbf{T}_U(\phi,\ 0,\ 0,\ 821)$	$\mathbf{T}_{R-}$	$\overline{p}_j \cdot \overline{t}_j$	$p_j \cdot t_j + \overline{p}_j \cdot \overline{t}_j$	$\overline{p}_j \cdot t_j + p_j \cdot \overline{t}_j$

For example, by choosing angles $\phi = [45°,\ 30°,\ 60°,\ 60°]$ and $\mathbf{T}_{R+}$ rotation matrix, the 13th value ($t = 13$) of the 6th *BF* ($p = 6$) can be obtained, using (2.3) and Table 5:

$$\begin{aligned} \mathbf{H}\,(5,\ 12) = \mathbf{H}\,(\{0101\}_2,\ \{1100\}_2) &= c_4 \cdot (-1) \cdot s_3 \cdot s_2 \cdot c_1 = \\ &= -\cos(60°) \cdot \sin(60°) \cdot \sin(40°) \cdot \sin(45°) = \\ &= -0.5 \cdot -0.866 \cdot 0.5 \cdot 0.707 \cdot = -0.1531, \end{aligned} \tag{2.4}$$

where $\{...\}_2$ – binary representation of indexes.

To generate all *BFs* using the described algorithm, more arithmetic operations are required than in the case when using the fast transform (FT). The coefficient of efficiency

Table 5: The details of example (2.4) [P1]

j-tais pakāpes koeficients	**Binārā izteiksme**	**Vērtība**
a_1 (LSB)	$1 \cdot 0 = 0$	$(-1)^0 = 1$
m_1 (LSB)	$1 \cdot 0 + 0 \cdot 1 = 0$	$(s_1)^0 = 1$
k_1 (LSB)	$0 \cdot 0 + 1 \cdot 1 = 1$	$(c_1)^1 = c_1$
a_2	$0 \cdot 0 = 0$	$(-1)^0 = 1$
m_2	$0 \cdot 0 + 1 \cdot 1 = 1$	$(s_2)^1 = s_2$
k_2	$1 \cdot 0 + 0 \cdot 1 = 0$	$(c_2)^0 = 1$
a_3	$1 \cdot 1 = 1$	$(-1)^1 = -1$
m_3	$1 \cdot 1 + 0 \cdot 0 = 1$	$(s_3)^1 = s_3$
k_3	$0 \cdot 1 + 1 \cdot 0 = 0$	$(c_3)^0 = 1$
a_4 (MSB)	$0 \cdot 1 = 0$	$(-1)^0 = 1$
m_4 (MSB)	$0 \cdot 1 + 1 \cdot 0 = 0$	$(s_4)^0 = 1$
k_4 (MSB)	$1 \cdot 1 + 0 \cdot 0 = 1$	$(c_4)^1 = c_4$

of arithmetic operations for this algorithm, by comparing to FT [P4]:

$$r_{ops} = \frac{M_{_M} + A_{_M}}{M_{_{FT}} + A_{_{FT}}} \tag{2.5}$$

If

$$r_{ops}(K_{max},\ N) = 1, \tag{2.6}$$

then the number of arithmetic operations is the same for both ways of generating *BFs*. Using (2.1), (2.2) and (2.6)„ the maximum number (K_{max}) of *BFs* can be obtained, for

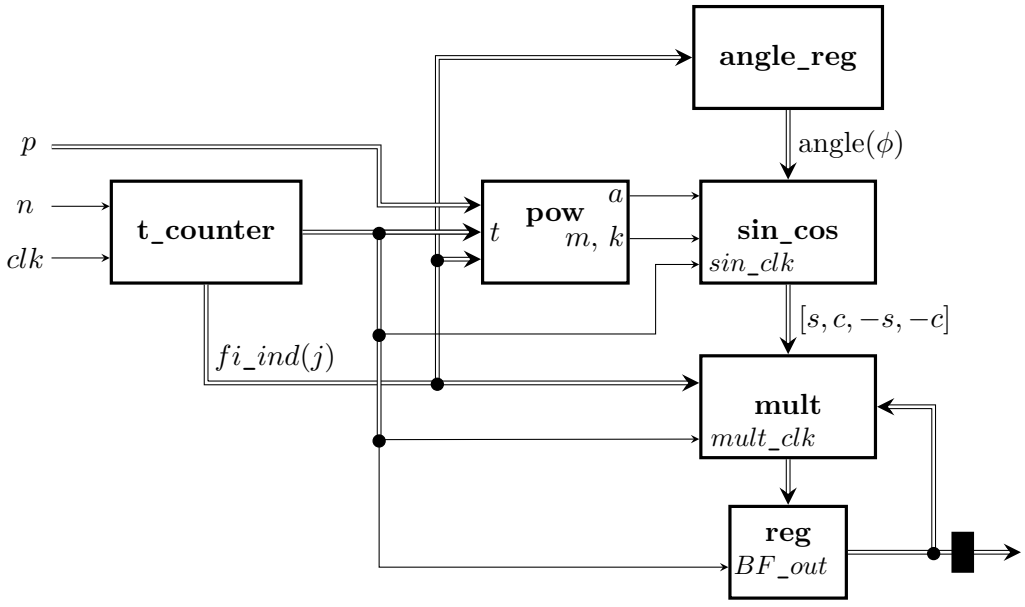


Figure 2.1: Simplified block diagram of the **CRAIMOT** *BF* generator [P1]

which individual generation, using the described algorithm, is as much efficient as using FT [P4]:

$$K_{max} = (3/\ln(2)) \cdot \ln(N)/(2 \cdot N - 1) \tag{2.7}$$

For practical use it is more convenient to use the following formula:

$$K_{max} \approx floor(1.44 \cdot \log_2(N) + 0.48), \tag{2.8}$$

where $floor$ – way of rounding of a number (see Section 3.2.1).

Using (2.3) and Table 4, a *BF* generator (see Figure 2.1) can be constructed, which each time calculates a single *BF* value. It enables even system operation and little memory consumption, and allows to generate long ($N = 2^{32}$) *BFs*. On the basis of such a generator we can build a simple signal synthesizer (see 2.2).

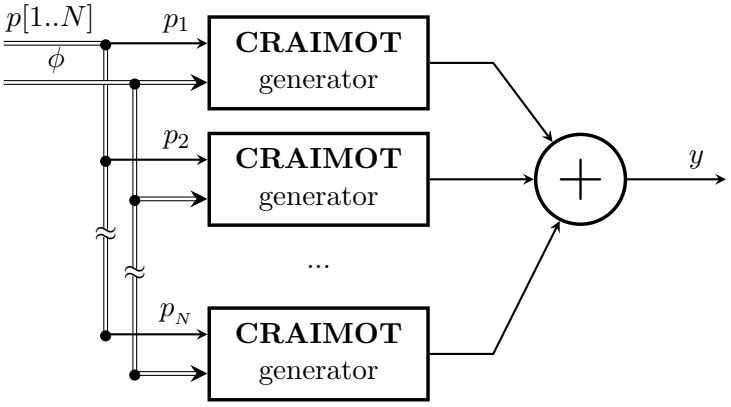


Figure 2.2: Simplified block diagram of the **CRAIMOT** signal synthesizer

Error estimation for *BF* generator The source of errors in the described generator lies with the method for calculation of *sin/cos* values. Māris Tērauds in his doctoral thesis [3] analyzed errors of *BFs* and *BF* partial sums in case *sin/cos* values are calculated using one of the two methods – linear interpolation (see Figure 2.3) or *CORDIC*. The experiments proved that *MSEs* of *BFs* and *BF* partial sums for an *FPGA*-based generator lay in the limits calculated by Tērauds [3], [P7].

The latest *FPGAs* have more and more internal memory, and, for that reason, the using of *sin/cos* table, which is the fastest method to get *sin/cos* values, becomes more effective. Since the estimation of errors arising when calculating *sin/cos* values using the table does not appear in the publications and Tērauds' Thesis [3], here follows a brief review on that. In the *sin/cos* table all angles $\alpha \in [0, 90°]$ (then the other sinuses and cosines can be determined using $\sin(\alpha)$) are evenly divided into 2^{n_α} elements, where n_α – the number of bits in the binary representation of angle. Sin values are stored in $Q1.x$

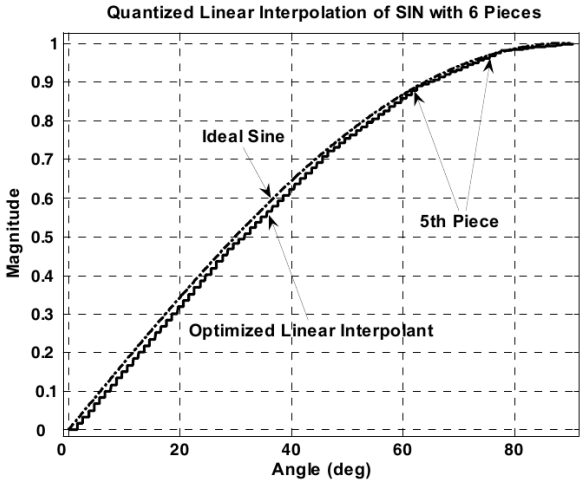


Figure 2.3: Optimized linear interpolation of sine for $\Delta_{max} \approx 0.004$ [P4]

fixed-point number format (x – *FPA WL*). Therefore, the *sin/cos* error depends on the angular error and the *FPA* error. The angular error (α_{err}) is larger if the required angle lays exactly in the middle between two angles specified in the table. The sine function is most sensitive to the change of angle when the value of angle is near zero. Consequently, the maximum error for sine angle in the table can be calculated. It is then scaled by the sine range defined in the table (if $\alpha \in [0,\ 90°]$, then $\sin(\alpha) \in [0, 1]$):

$$\alpha_{err} = \frac{90}{2 \cdot 2^{n_\alpha}}, \quad \epsilon_{\alpha\%} = \sin(\alpha_{err}) \cdot 100\% \tag{2.9}$$

The extent of *FPA* error depends on the wordlength of sine values and on the particular rounding method being used (see Section 3.5). Usually, when building a sine table, the most precise rounding (*round*) is used. The amount of memory (in bits) needed for sine table is calculated as follows:

$$N_{bits} = 2^{n_\alpha} \cdot x, \tag{2.10}$$

where 2^{n_α} - the number of table elements, x - $Q1.x$ *FPA WL*.

2.2.2 CRAIMOT spectrum analyzer [P2]

A simple spectrum analyzer (see Figure 2.4) can be constructed on the basis of *BF* generator. It allows to calculate a particular spectrum coefficient using scalar product of signal and corresponding *BF*. It requires some time to calculate all the spectrum coefficients, because incoming input signal samples are stored in the signal register and then N-times the scalar multiplication of vectors is performed, that is, the signal vector is multiplied by all *BFs*.

Operations for spectrum calculation in the analyzer described in this section are performed sequentially in time, and therefore its operation speed is rather low. Section 2.3

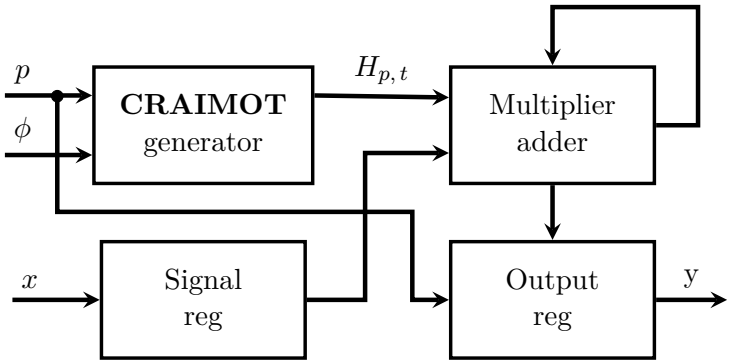


Figure 2.4: Simplified block diagram of the **CRAIMOT** spectrum analyzer [P2]

deals with a tree-like algorithm for calculating signal spectrum. Its operation speed is much faster, because all spectrum coefficients are calculated concurrently.

Summary on papers [P1], [P2], [P4]

Obtained results

- A serial architecture for **CRAIMOT** generator, based on *sin/cos* products, is created. An architecture for the corresponding signal synthesizer is created.
- A serial architecture for **CRAIMOT** spectrum analyzer is created.
- Three different algorithms for obtaining *sin/cos* values have been examined.

Conclusions

- Generation of a small number of *BFs* (2.7),(2.8), using *sin/cos* products, is more effective than using the fast algorithm. For example, when $N = 64$, using the serial algorithm to generate less than 9 *BFs* requires a smaller number of arithmetic operations than using the fast transform.
- Both the amount of errors and necessary resources (in *FPGA* – logical elements, memory bits, *DSP* elements) depend on the method used to calculate *sin/cos* values.
- A signal spectrum analyzer, based on *BF* generator, can be easily constructed, but it has a low operation speed. For each spectrum value the scalar multiplication of N-sample long vectors must be executed, one at a time.

2.3 Tree-like structure of signal decomposition/reconstruction system

A tree-like structure allows to perform transforms using pipelining [19], which utilizes computing resources (in this case, *FPGA*) very effectively.

2.3.1 Signal decomposition [P10],[P11]

This section deals with the correspondence between the matrix form and the tree-like structure of signal decomposition system. Figure 2.5 shows a simplified block diagram of

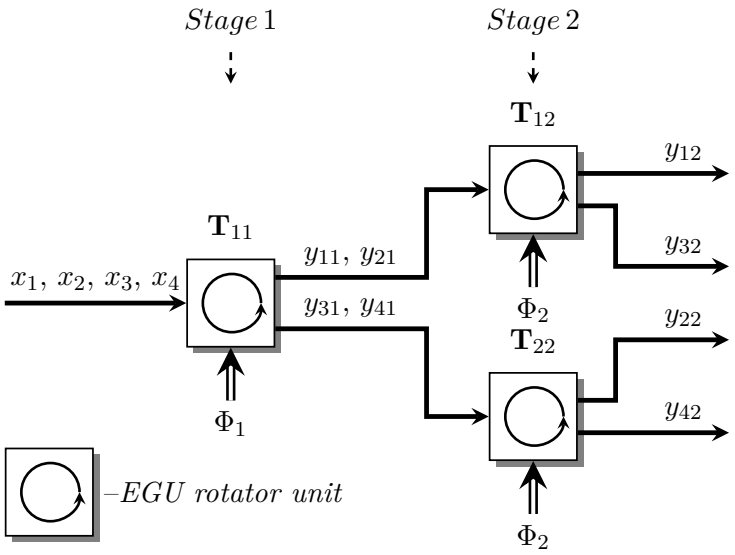


Figure 2.5: **CCRAIMOT** *DE* (for N=4) simplified block diagram [P11]

the tree-like structure of signal decomposition. Parameters for rotator units can change in time, and thus the spectrum of **CRABOT** class is obtained. A timing diagram in Figure 2.6 shows changing of parameters ($N = 4$). And a tree-like structure in Figure 2.7 also shows changing of parameters ($N = 8$). Although each structure of rotation matrix has its own rotator unit, these units can be easily obtained using the automatic algorithm described in section 3.6.

A 4-sample signal spectrum in matrix form can be obtained using fast **CCRAIMOT** transform – two stairs-like matrices with the same structure, where each matrix contains

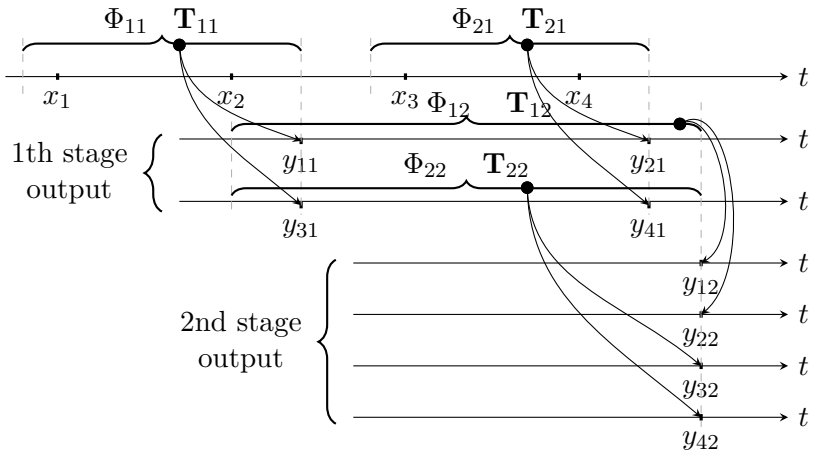


Figure 2.6: Simplified timing diagram [P11]

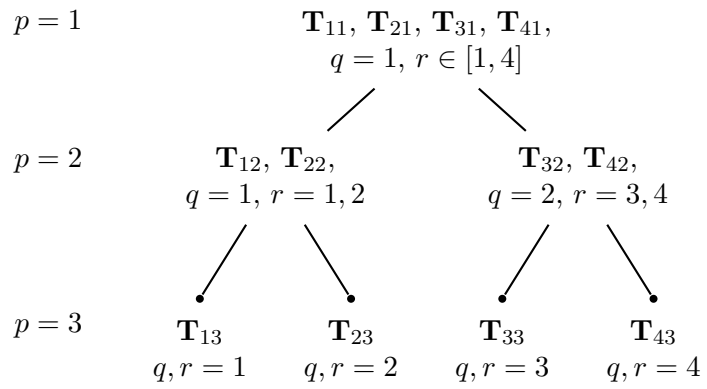


Figure 2.7: Decomposition tree structure and addressing example $N = 8$ [P11]

the same rotation parameters:

$$\mathbf{Y}_4 = \mathbf{B}_s\left(\Phi_2\right) \cdot \mathbf{B}_s\left(\Phi_1\right) \cdot \mathbf{X}_4 \tag{2.11}$$

Written with all matrix elements, the product of the signal and the first matrix is:

$$\begin{aligned} \mathbf{Y}_1 = \mathbf{B}_s(\Phi_1) \cdot \mathbf{X} &= \begin{bmatrix} c_1 e^{i} & s_1 e^{-i} & 0 & 0 \\ 0 & 0 & c_1 e^{i} & s_1 e^{-i} \\ -s_1 e^{i} & c_1^{-i} & 0 & 0 \\ 0 & 0 & -s_1 e^{i} & c_1^{-i} \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \\ &= \begin{bmatrix} c_1 e^{i} \cdot x_1 + s_1 e^{-i} \cdot x_2 \\ c_1 e^{i} \cdot x_3 + s_1 e^{-i} \cdot x_4 \\ c_1 e^{i} \cdot x_2 - s_1 e^{-i} \cdot x_1 \\ c_1 e^{i} \cdot x_4 - s_1 e^{-i} \cdot x_3 \end{bmatrix} = \begin{bmatrix} y_{11} \\ y_{21} \\ y_{31} \\ y_{41} \end{bmatrix} \end{aligned} \tag{2.12}$$

When implementing signal decomposition in the pipeline architecture, decomposition of 4-sample signal requires two stages. The first of them is shown in Figure 2.8.

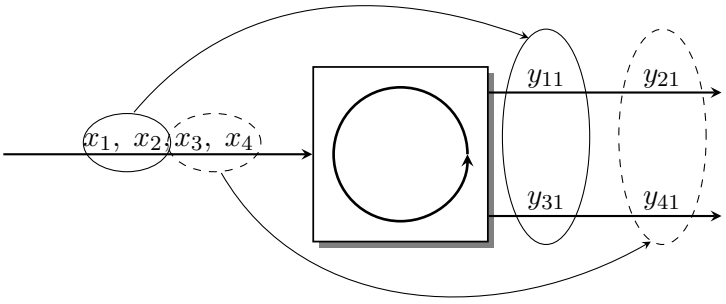


Figure 2.8: First stage of decomposition of 4-sample signal

Samples enter the rotator input consecutively, and in the same order they are then in pairs processed. From two consecutively incoming samples two parallel intermediate results having twice as low frequency are produced. Mathematically, it is the product of two independent 2×2 ($\mathbf{T}_U$) rotation matrices and vectors $\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$ and $\begin{bmatrix} x_3 \\ x_4 \end{bmatrix}$, or the product of 4-sample long signal and block-like rotation matrix (1.15):

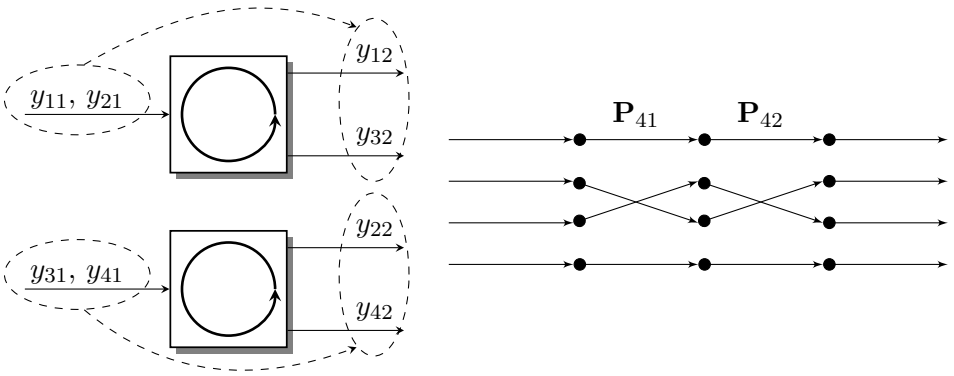
$$\mathbf{Y}_{b1} = \mathbf{B}_b(\Phi_1) \cdot \mathbf{X} = \begin{bmatrix} c_1 e^{i} & s_1 e^{-i} & 0 & 0 \\ -s_1 e^{i} & c_1^{-i} & 0 & 0 \\ 0 & 0 & c_1 e^{i} & s_1 e^{-i} \\ 0 & 0 & -s_1 e^{i} & c_1^{-i} \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \\ = \begin{bmatrix} c_1 e^{i} \cdot x_1 + s_1 e^{-i} \cdot x_2 \\ c_1 e^{i} \cdot x_2 - s_1 e^{-i} \cdot x_1 \\ c_1 e^{i} \cdot x_3 + s_1 e^{-i} \cdot x_4 \\ c_1 e^{i} \cdot x_4 - s_1 e^{-i} \cdot x_3 \end{bmatrix} = \begin{bmatrix} y_{11} \\ y_{31} \\ y_{21} \\ y_{41} \end{bmatrix} \tag{2.13}$$

Then the intermediate result and the second stairs-like matrix can be multiplied (2.14). The sequence of spectrum obtained in such a way will be regarded as correct.

$$\mathbf{Y} = \mathbf{B}_s(\Phi_2) \cdot \mathbf{Y}_1 = \begin{bmatrix} c_2 e^{i} & s_2 e^{-i} & 0 & 0 \\ 0 & 0 & c_2 e^{i} & s_2 e^{-i} \\ -s_2 e^{i} & c_2^{-i} & 0 & 0 \\ 0 & 0 & -s_2 e^{i} & c_2^{-i} \end{bmatrix} \cdot \begin{bmatrix} y_{11} \\ y_{21} \\ y_{31} \\ y_{41} \end{bmatrix} = \\ = \begin{bmatrix} c_2 e^{i} \cdot y_{11} + s_2 e^{-i} \cdot y_{21} \\ c_2 e^{i} \cdot y_{31} + s_2 e^{-i} \cdot y_{41} \\ c_2 e^{i} \cdot y_{21} - s_2 e^{-i} \cdot y_{11} \\ c_2 e^{i} \cdot y_{41} - s_2 e^{-i} \cdot y_{31} \end{bmatrix} = \begin{bmatrix} y_{12} \\ y_{22} \\ y_{32} \\ y_{42} \end{bmatrix} \tag{2.14}$$

The second stage of pipelined decomposition is shown in Figure 2.9a. The second stage consists of two complex rotators. Similarly to the first stage (see Figure 2.8), from two consecutive samples at input two parallel samples are produced at output. To obtain both input samples (y_{11} and y_{21}) for the first complex rotator in the second stage all of the four input signal samples (x_1, x_2, x_3, x_4) (2.13) are required. The same applies to the second rotator.

It can be considered that by joining complex rotators in a cascade of stages, a permutation of samples in intermediate results is performed (similarly to (1.17), (1.18)). Figure 2.9b shows a graphical depiction of permutation matrices for $N = 4$. In the second stage four samples are rotated in pairs using two separate rotators, and it is expressed as follows:



(a) The second stage

(b) Graphical depiction of permutation matrices ($N = 4$)

Figure 2.9: Decomposition of 4-sample signal

$$\mathbf{Y}_b = \mathbf{B}_b(\Phi_2) \cdot \mathbf{P}_4 \cdot \mathbf{Y}_{b1} = \begin{bmatrix} c_2 e^{i} & s_2 e^{-i} & 0 & 0 \\ -s_2 e^{i} & c_2^{-i} & 0 & 0 \\ 0 & 0 & c_2 e^{i} & s_2 e^{-i} \\ 0 & 0 & -s_2 e^{i} & c_2^{-i} \end{bmatrix} \cdot \begin{bmatrix} y_{11} \\ y_{21} \\ y_{31} \\ y_{41} \end{bmatrix} =$$

$$= \begin{bmatrix} c_2 e^{i} \cdot y_{11} + s_2 e^{-i} \cdot y_{21} \\ c_2 e^{i} \cdot y_{21} - s_2 e^{-i} \cdot y_{11} \\ c_2 e^{i} \cdot y_{31} + s_2 e^{-i} \cdot y_{41} \\ c_2 e^{i} \cdot y_{41} - s_2 e^{-i} \cdot y_{31} \end{bmatrix} = \begin{bmatrix} y_{12} \\ y_{32} \\ y_{22} \\ y_{42} \end{bmatrix} \tag{2.15}$$

Using expressions (2.15) and (2.13), the results of the structure in Figure 2.5 are produced in an order that is most convenient for checking decomposition system operation (no permutations are needed).

Increasing the number of stages n increases the number of samples to be processed and

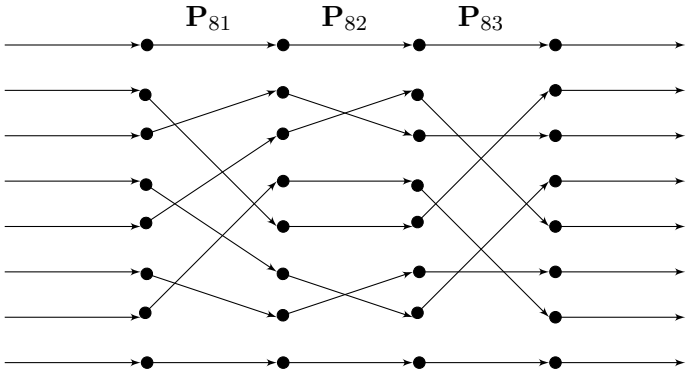


Figure 2.10: Graphical depiction of permutation matrices matrices ($N = 8$)

changes the structure of permutation matrices. In the case of 4-samples, both permutation matrices are equal, in all the other cases (for $N > 4$) all permutation matrices are different. Figure 2.10 shows a graphical depiction for $N = 8$.

Using permutation matrices, it is possible to obtain intermediate results of transform implemented using the pipeline approach (by using sparse rotation matrices with a block-like structure), after each stage. Intermediate results being calculated mathematically essentially facilitates performing of parametrical transform. To obtain block-like rotation matrices, the Kronecker product can be used:

$$\mathbf{B}_b(\Phi_1) = \mathbf{I}_4 \otimes \mathbf{T}_U(\Phi_1, ID) = \left[\begin{array}{cccc} \mathbf{T}_U(\Phi_1, ID) & \mathbf{0}_2 & \mathbf{0}_2 & \mathbf{0}_2 \\ \mathbf{0}_2 & \mathbf{T}_U(\Phi_1, ID) & \mathbf{0}_2 & \mathbf{0}_2 \\ \mathbf{0}_2 & \mathbf{0}_2 & \mathbf{T}_U(\Phi_1, ID) & \mathbf{0}_2 \\ \mathbf{0}_2 & \mathbf{0}_2 & \mathbf{0}_2 & \mathbf{T}_U(\Phi_1, ID) \end{array}\right]\Bigg|_{ID=424} =$$

$$= \left[\begin{array}{cccccccc} c_1 e^{i} & s_1 e^{-i} & 0 & 0 & 0 & 0 & 0 & 0 \\ -s_1 e^{i} & c_1 e^{-i} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & c_1 e^{i} & s_1 e^{-i} & 0 & 0 & 0 & 0 \\ 0 & 0 & -s_1 e^{i} & c_1 e^{-i} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & c_1 e^{i} & s_1 e^{-i} & 0 & 0 \\ 0 & 0 & 0 & 0 & -s_1 e^{i} & c_1 e^{-i} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & c_1 e^{i} & s_1 e^{-i} \\ 0 & 0 & 0 & 0 & 0 & 0 & -s_1 e^{i} & c_1 e^{-i} \end{array}\right] \tag{2.16}$$

Using permutation matrix P_{81}, a stairs-like rotation matrix can be obtained from a sparse block-like rotation matrix. The sequence of spectrum coefficients $\mathbf{Y}_b$ calculated during signal decomposition with pipelining differs from that which is obtained using three stairs-like factorized matrices. It applies also for $N = 8$. $\mathbf{Y}_b$ is calculated using $\mathbf{P}_{81}$ and $\mathbf{P}_{82}$:

$$\mathbf{Y}_b = \mathbf{B}_b(\Phi_3) \cdot \mathbf{P}_{82} \cdot \mathbf{B}_b(\Phi_2) \cdot \mathbf{P}_{81} \cdot \mathbf{B}_b(\Phi_1) \cdot \mathbf{X} \tag{2.17}$$

Due to simple notation, this section discusses orthogonal transforms of one particular class (**CCRAIMOT**) only. In general case (**CRABOT** class), a factorized block-like rotation matrix (for $N = 4$) is expressed, using Kronecker product [3], as follows:

$$\begin{aligned} \mathbf{B}_b(\mathbf{\Phi}_k) &= \{\mathbf{I}_2\} \otimes \{\mathbf{T}_{U,1k}(\Phi_{1k}, ID_{1k}), \mathbf{T}_{U,2k}(\Phi_{2k}, ID_{2k})\} = \\ &= \left[\begin{array}{cc} \mathbf{T}_{U,1k}(\Phi_{1k}, ID_{1k}) & \mathbf{0}_2 \\ \mathbf{0}_2 & \mathbf{T}_{U,2k}(\Phi_{2k}, ID_{2k}) \end{array}\right] \end{aligned} \tag{2.18}$$

Note that a full orthogonal transform requires $n = log_2(N)$ sparse rotation matrices.

2.3.2 Signal reconstruction

Figure 2.11 shows a simplified block diagram of **CCRAIMOT** reconstruction. While for decomposition parallel outgoing samples are obtained from consecutively incoming samples (fan-out tree-like structure), in case of reconstruction there are parallel incoming samples and consecutively outgoing samples (fan-in tree-like structure). Besides, for each *RE* block the discretization frequency of consecutive samples is twice as high as that of parallel samples (independent of the fact whether consecutive samples are at the input or the output of *RE*).

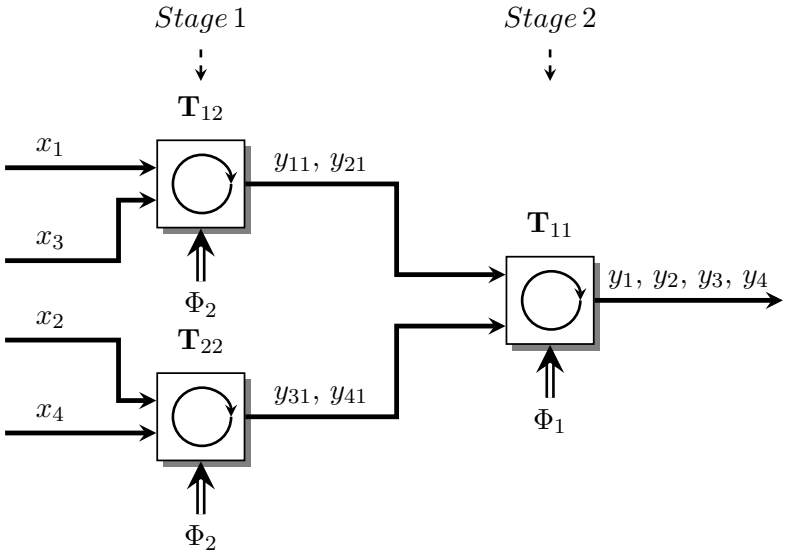


Figure 2.11: Simplified block diagram of **CCRAIMOT** reconstruction, $N = 4$

Summary

Obtained results

- Analysis of the structure and operation of multilevel parametrical tree-like signal decomposition/reconstruction system [P10], [P11].

Conclusions

- Each structure $\mathbf{T}_U(\phi, \psi, \gamma, ID)$ of elementary rotation matrix has its own corresponding *RE*.
- *RE* performs a rotation of two-dimensional signal vector, therefore samples are fed to the *RE* input in pairs – as $N = 2$-sample long vectors.
- System reconfiguration is done by changing *RE* parameters.

- There are at least two possible architectures how to implement *RE* – the traditional algorithm and the *CORDIC* algorithm (see Section 3.3).
- In a tree-like signal decomposition system, after each a decimation is performed (the clock frequency reduces twice). It allows to reduce sample processing time.
- In a tree-like signal reconstruction system, after each stage the upsampling is performed (the clock frequency increases twice). It means that at the last stage samples must be processed at the same rate as at the first stage of decomposition system.

2.4 Signal decomposition using complex *FIR* filters

Signal decomposition using *RE* described in previous section can be implemented using complex *FIR* filters, too. So, at first, this section deals with the parameters for complex *FIR* filters obtained from the rotation matrix (1.3). After that, the design of decomposition/reconstruction system is presented and its advantages and disadvantages in comparison to that described in Section 2.3 assessed.

2.4.1 2.4.1. Transfer functions of complex filters

Orthogonal filters can be used to decompose an input signal into components. In wavelet processing, each pair of orthogonal filters decomposes a signal into the approximation component (low frequencies) and the detail component (high frequencies). Orthogonal filters are used not only for wavelets, but also, for example, when performing the Hilbert transform [20], where complex and real *FIR* filters are joined into a single orthogonal filter system.

Complex orthogonal filters and their properties are discussed in [8]. Time-varying real filter banks for different structures are discussed in [21], [22] and [23] and in many other papers. The transfer function of digital filters [24] is expressed as follows:

$$H(z) = \frac{b_0 + b_1 z^{-1} + \cdots + b_N z^{-N}}{1 + a_1 z^{-1} + \cdots + a_N z^{-N}} \tag{2.19}$$

Given the transfer function, the frequency transfer response of a filter can be calculated. Since *FIR* (finite impulse response) filters are used, the frequency transfer response is calculated as follows:

$$H(\omega) = \sum_{k=0}^{M-1} b(k) e^{-j\omega k} \tag{2.20}$$

The frequency transfer response is complex. Its modulus (usually, in decibels) is named as the filter magnitude-frequency response. The filter phase-frequency response is calculated as follows:

$$\Theta(\omega) = \tan^{-1}\left(\frac{\mathfrak{Im}H(\omega)}{\mathfrak{Re}H(\omega)}\right) \tag{2.21}$$

The complex rotation matrix (1.6) can be interpreted as two filters (followed by downsampling), where the first row of the matrix corresponds to the approximation filter **Lo** in Figure 1.1 and the second row – to the detail filter **Hi**. Using (2.20) and (2.21), the magnitude- and the phase-frequency transfer response for both filters can be calculated. For example, the frequency transfer response of **Lo** filter is calculated as:

$$\begin{aligned} H(\omega) = b(1) + b(2)e^{-j\omega} = \cos(\phi)e^{j\gamma} + \sin(\phi)e^{-j\psi}e^{-j\omega} = \\ = \cos(\phi)e^{j\gamma} - \frac{\sin(\phi)e^{j\psi}}{e^{j\omega}}, \end{aligned} \tag{2.22}$$

where $\omega \in [0, \pi]$, (ϕ, ψ, γ) – filter parameters

For arbitrarily chosen filter parameters $\Phi_k = (\phi_k, \psi_k, \gamma_k) = \left(\frac{\pi}{4}, \frac{\pi}{3}, \frac{\pi}{4}\right)$, the magnitude- and the phase-frequency response curves[7] of the respective approximation and detail filters are shown in Figure 2.12.

Using the transposed matrix (1.12), the reconstruction filters can be obtained (see Figure (2.13).

[7]Frequency values (x-axis) are scaled by $\frac{Fs}{2}$

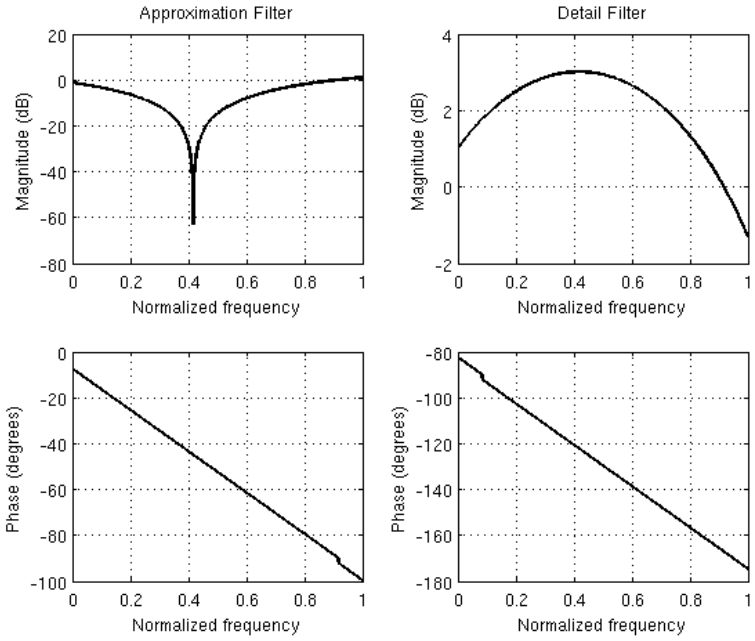


Figure 2.12: Magnitude- and Phase-frequency response curves of the approximation and detail filters (**De**compozition), $\phi = \frac{\pi}{4}$, $\psi = \frac{\pi}{3}$, $\gamma = \frac{\pi}{4}$

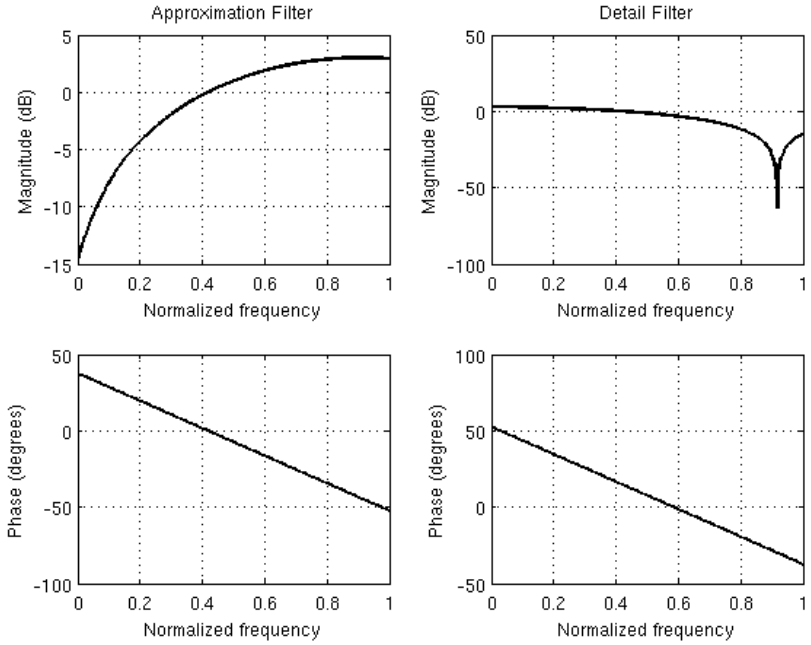


Figure 2.13: Magnitude- and Phase-frequency response curves of the approximation and detail filters (**Re**construction), $\phi = \frac{\pi}{4}$, $\psi = \frac{\pi}{3}$, $\gamma = \frac{\pi}{4}$

By joining into a single system the filter pair obtained from the direct rotation and the filter pair obtained from the transposed matrix, a decomposition-reconstruction system (see Figure 2.14) can be built, which allows to reconstruct a signal fully. To achieve a lossless reconstruction, the following condition must hold [22]:

$$\mathbf{Lo_D}^T\mathbf{Hi_D} + \mathbf{Lo_R}^T\mathbf{Hi_R} = \mathbf{I} \tag{2.23}$$

By changing parameters, it is possible to obtain a practically unlimited number of various real and complex first-order *FIR* filter pairs (see Figure 2.15). In the case of complex filters (Figure 2.15b), some common trends can be identified in the dependence of magnitude-frequency response[8] on the angles, that is, parameters:

- Magnitude-frequency response curves for complex filters are much more complicated as those for real filters.

[8]Note that filtering is followed by downsampling

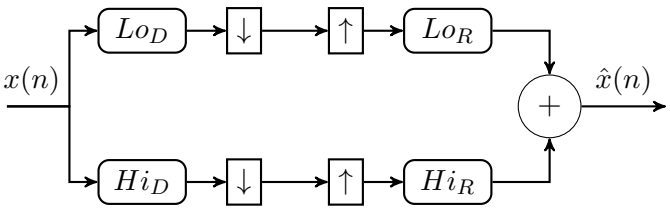
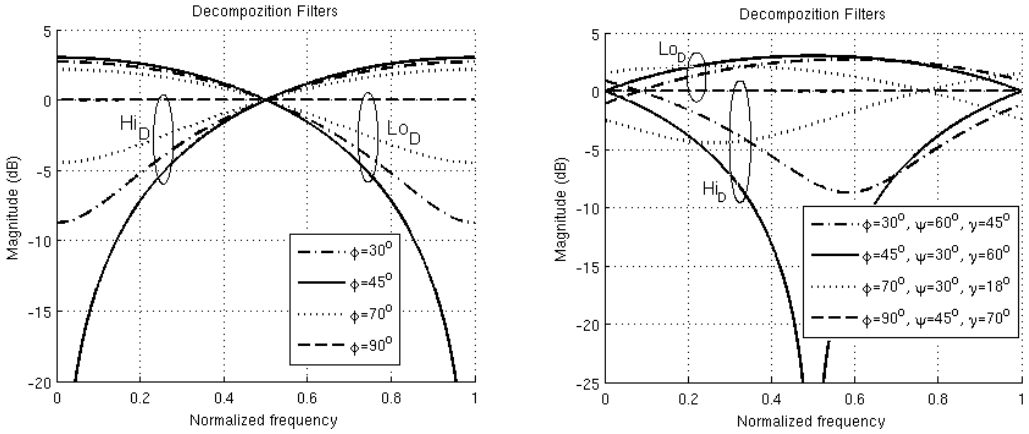


Figure 2.14: De–Re filters

- Magnitude-frequency response curves for complex approximation/detail filters (**Lo**/**Hi**) can take the form of band-pass or band-stop filter, depending on the elementary rotation matrix *ID*.
- The angle ϕ determines the bandwidth of band-stop filters. However, there are no exact formulas yet.
- The angles ψ and γ determine the pass-band/stop-band frequency of band-pass/band-stop filters. However, there are no exact formulas yet.



(a) Real filters ($\mathbf{T}_U(\phi, 0, 0, 812)$)

(b) Complex filter ($\mathbf{T}_U(\phi, \psi, \gamma, 812)$)

Figure 2.15: Magnitude response curves of the approximation and detail filters (**De**composition), taken partly ((a)) from [P3], [P6] un [P8])

2.4.2 Cascading of complex orthogonal filters

Let's construct a complex *FIR* filter (Figure 2.16). Its simplified version (using real coefficients calculated inside the filter) is used in four different roles in a Haar-like decomposition-reconstruction system shown in Figure 2.17 – both the approximation filter (AD) and the detail filter (DD) of decomposition, and both the approximation filter (AR) and the detail

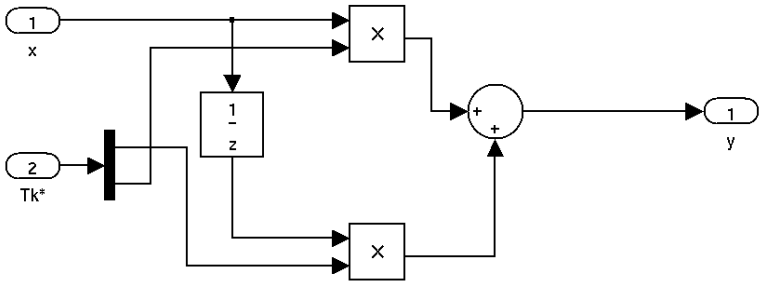


Figure 2.16: Simplified *Simulink* model of complex *FIR* filter

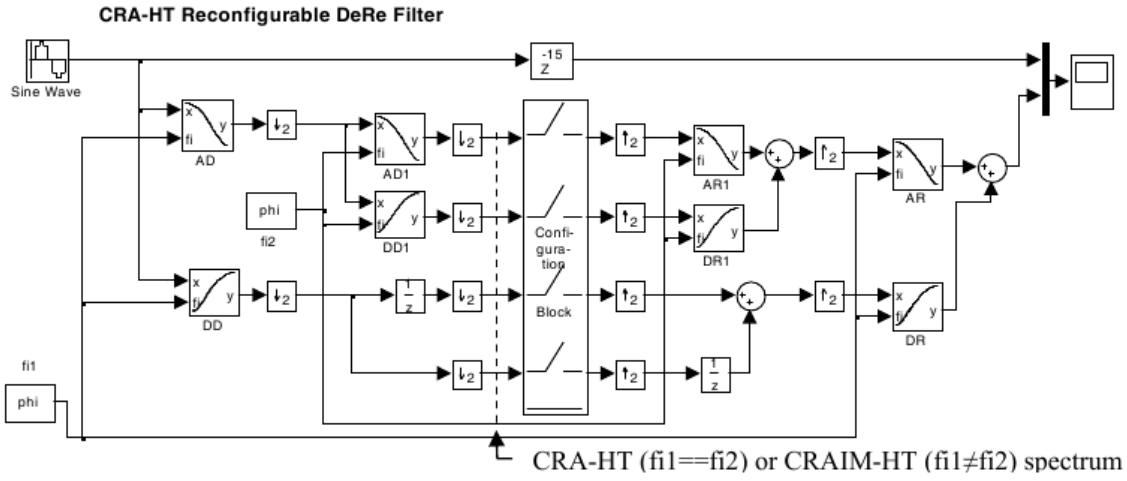


Figure 2.17: The **RA-HT** *DeRe* filter block diagram in *Simulink* [P3]

filter (DR) of reconstruction. To implement complex *FIR* filters, the direct-form digital filter structure is used [26]. Note that in this system an input signal and all filter coefficients are complex quantities, and their mutual multiplication is a complicated operation [27].

However, the main disadvantage of such a system is a complicated tuning of the filters, because filter coefficients are calculated from three real parameters – angles ϕ, ψ and γ. In the system in Figure (2.17 the matrix $\mathbf{T}_U(\phi, \psi = 0, \gamma = 0, 424)$ is used to obtain all filter coefficients. Depending on the structure of complex matrix (1.3), different filter coefficients can be obtained. A diagram for the calculation of filter coefficients for a particular structure is shown in Figure 2.18.

An advantage of such a system is that it does not change for different structures of

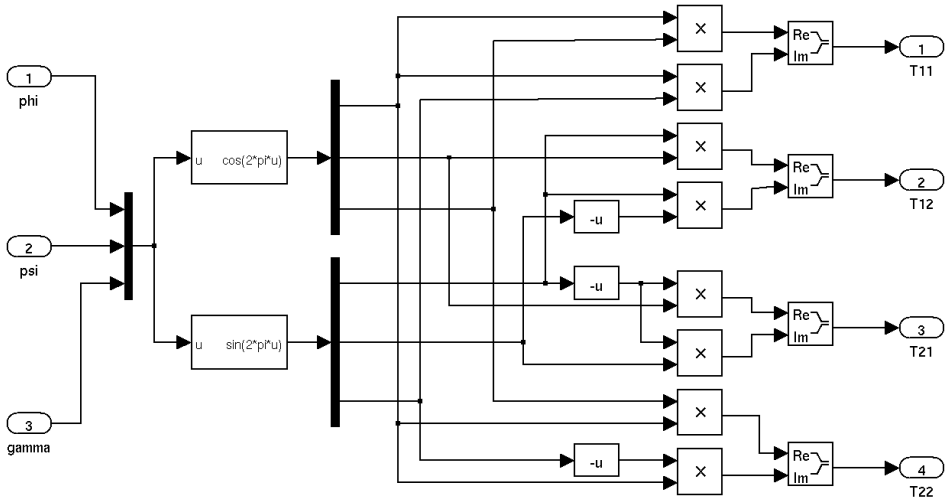


Figure 2.18: Simulink model for obtaining the complex matrix $\mathbf{T}_U(\phi, \psi, \gamma, 424)$ (filter coefficients)

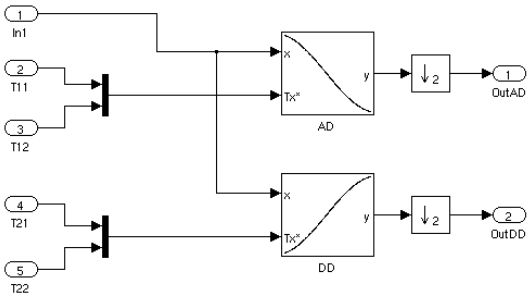


Figure 2.19: *Simulink* model of the cascading of decomposition stages, using two *FIR* filters

complex rotation matrices – only the filter coefficients vary. However, as seen in Figure 2.18, calculating the coefficients for orthogonal filters is a relatively complicated task, and a tunable system built in such a way cannot be regarded as optimal.

When building a decomposition-reconstruction system using filters, the downsampling must be taken into account. Using two *FIR* filters, the sampling frequency must be decreased twice. Figure 2.19 shows a *Simulink* model of the first stage of signal decomposition. The corresponding Simulink timing diagrams, if real part of input signal is sinusoidal, but imaginary – saw-like, are shown in Figure 2.20.

For signal reconstruction, the transposed matrix of the complex matrix $\mathbf{T}_U$ is used – the complex conjugates of the coefficients are used, the rows and columns are swapped, and the sampling frequency must be increased twice (upsampling). Figure 2.21 shows a *Simulink* model of signal reconstruction, where *In1* is the signal *outAD* and *In2* is the signal *outDD* in the decomposition model.

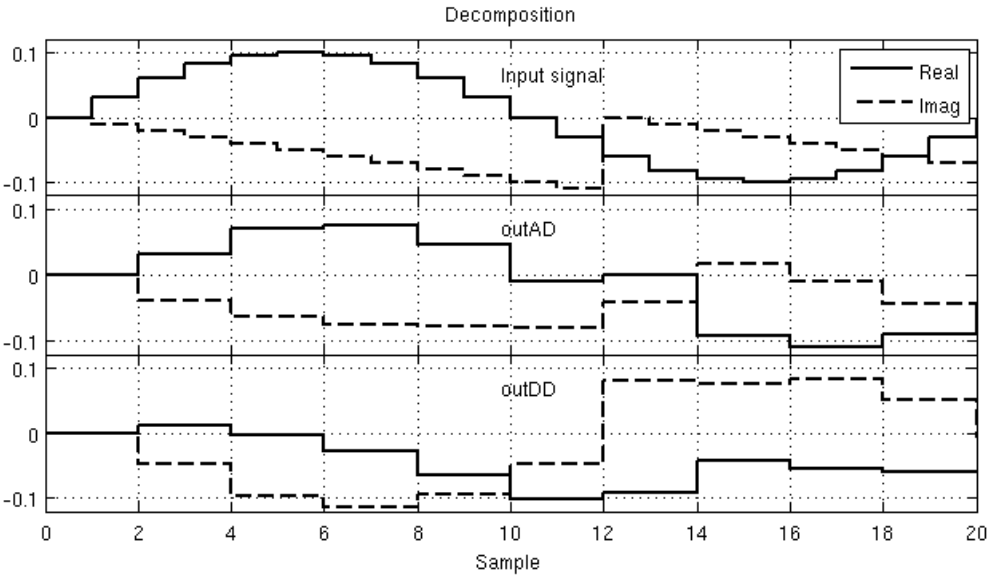


Figure 2.20: *Simulink* timing diagrams of input signal decomposition, $[\phi = \frac{\pi}{4}, \psi = \frac{\pi}{3}, \gamma = \frac{\pi}{6}]$

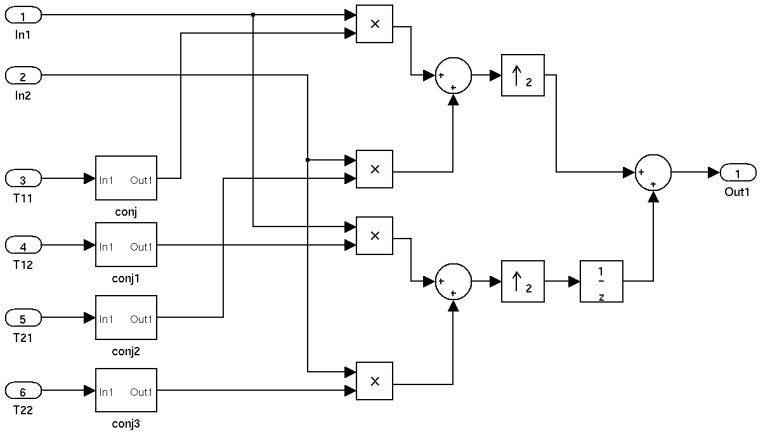


Figure 2.21: *Simulink* model of signal reconstruction

Summary

Obtained results

- Provisional analysis of suitability of the alternative decomposition-reconstruction system based on *FIR* filters for implementation in *FPGA*.

Conclusions

- The structure of filters does not change by changing system parameters.
- The system operates with sequential input data – input samples need not to be divided into N-sample blocks.
- All first order complex filters consist of 16 multipliers and 12 adders, not allowing to perform simplifications.
- In the case of complex filters, each coefficient of both filters (**Lo** and **Hi**) depends on all three angles (parameters). This complicates the tuning of the system and makes it resource-intensive.
- The coefficients of real filters depend only on one angle ϕ, making the system relatively easy tunable.
- In the further research on 1-D signal transforms complex quantities are being used. Therefore, in view of previous conclusions, signal decomposition using complex filters is, for the time being, assumed to be inefficient.
- It can be considered that also in the decomposition/reconstruction using *RE*, the mentioned manipulations with input signal – filtering using **Lo** and **Hi** filters and downsampling[9] – are performed.

[9] for reconstruction, increasing the sampling frequency, respectively

3 FPGA Implementation of the Phi-Transform Algorithms

The implementation of the algorithms described in Chapters 1 and 2 into *FPGA* chips is an essential part of this research. This chapter deals with different methods of implementation and the problems associated with them. After considering the advantages and disadvantages of the architectures described in Chapter 2, it has been decided to investigate further signal decomposition-reconstruction systems using *RE*.

3.1 Floating- and fixed-point arithmetic

Although floating-point arithmetic is being increasingly implemented in *FPGA* chips [35]-[38], in the algorithms proposed in this research fixed-point arithmetic (*FPA*) is used. Further, both these methods of representing and calculating real numbers will be supported [39]. But the use of floating-point arithmetic does not solve all precision problems [40], and it complicates, for example, the simulation of *VHDL* code.

Algorithm implementation using fixed-point arithmetic involves fixed-point errors [41]. The estimation of fixed-point errors for *RE* is described in Section 3.5.

3.2 *FPGA* implementation of *DSP* elements

3.2.1 Multiplier with variable output wordlengths (*WL*)

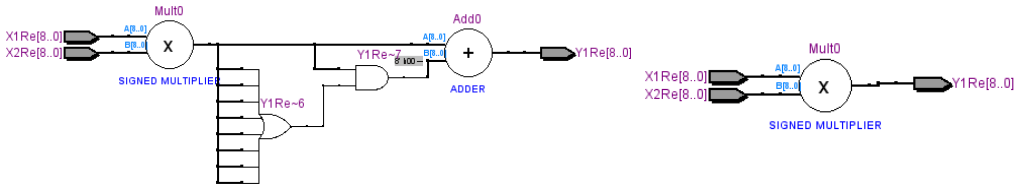
When performing a multiplication in *FPGA* using fixed-point arithmetic (practically all the latest generations of Altera *FPGAs* have fixed-point multipliers [42]), output signal has a twice as large *WL* as in input. By implementing algorithms consisting of several multipliers, it would create numerous problems because of excessive increasing of signal *WLs*. Usually, a scaling is done, which includes bit truncation. In *Q1.x* arithmetic [43], when truncating the number of bits to w bits (truncating the lowest bits), the precision of the result decreases. It is determined by the quantization step [43]:

$$\epsilon_x = \frac{1}{2^x} \tag{3.1}$$

where x – the number of fractional bits in $Q1.x$ *FPA* ($x = w–1$).

Bit truncation can be performed using four different rounding modes:

- floor – the result is rounded down towards $-\infty$ till equivalent integer,
- fix – the result is rounded toward zero till equivalent integer,
- ceil – the result is rounded up towards $+\infty$ till equivalent integer,
- round – the result is rounded to nearest equivalent integer.



(a) *fix* rounding

(b) *floor* rounding

Figure 3.1: *RTL* description of multipliers using bit truncation, generated by *HDL Coder*

Each of these rounding modes is implemented in a different way, and, consequently, the number of additionally required logic elements differs as well. Using *Simulink HDL coder* and **fi()** object in *MatLab*, it is easy to choose the most suitable rounding mode. Figure 3.1 shows a hardware model of *fix* and *floor* rounding for bit truncation from 16 to 8 bits. *Floor* is the only rounding mode that does not require additional logic elements (see Table 7).

3.2.2 Serial-to-parallel and parallel-to-serial converters

For automation of synthesis of different rotator structures, arithmetic operations are defined as expressions – matrix transforms and it means that two samples must be taken at the input simultaneously. For that reason, in real devices, a converter is needed before

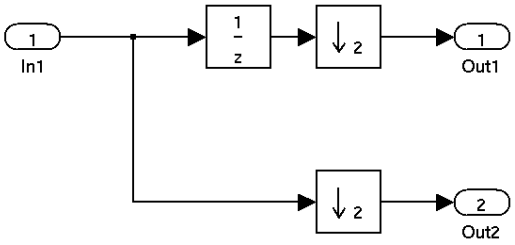


Figure 3.2: Serial-to-parallel converter (*Simulink* block diagram)

each rotator, which converts data incoming in series into data occurring simultaneously. The code for such converter could be directly written in *VHDL*, however, it is more convenient and visually understandable to use *Simulink* and *HDL Coder*, in order to obtain the *VHDL* code corresponding to the block diagram in Figure 3.2.

Table 6: The number of logic elements

	S2P	P2S
LUT's	50	49
ALUT's	34	18

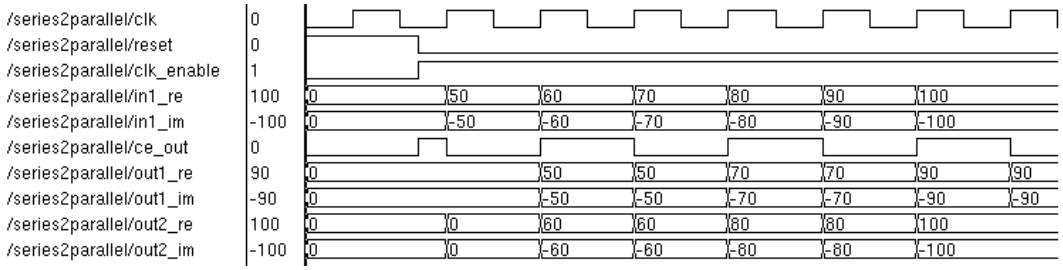


Figure 3.3: Timing diagram of the serial-to-parallel converter

A *ModelSim* timing diagram of the generated *VHDL* code is shown in Figure 3.3. *ModelSim* is a widely used tool for *HDL* code simulation [28]-[34]. A more detailed description of *VHDL* code generation using *Simulink HDL Coder* is given in Section 3.6. In addition to the complex input and two outputs, the *HDL coder* adds standard digital device outputs – clock pulse *clk*, operation enabling *clk_enable* and *reset*.

A parallel-to-serial data conversion is done in a similar way – Figure 3.4 shows a *Simulink* block diagram to perform the opposite operations. A *ModelSim* timing diagram

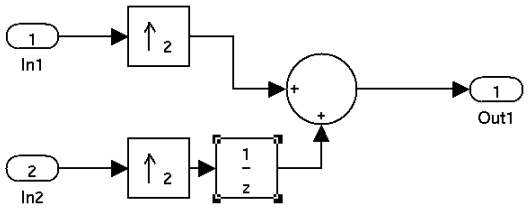


Figure 3.4: Parallel-to-serial converter (*Simulink* block diagram)

of the parallel-to-serial converter is shown in Figure 3.5.

Since there are different programmable elements for different *FPGA* families, the number of logic elements required for one and the same algorithm may vary. On *Altera FPGAs* two kinds of programmable logic elements are possible: LUT – Look Up Table and $ALUT$ – Adaptive Look-Up Table [44]. The advantage of $ALUT$ is that these elements can be combined in groups. Table 6 shows the number of *LUTs* and *ALUTs* required for implementation of serial-to-parallel (*S2P*) converter and parallel-to-serial (*P2S*) converter.

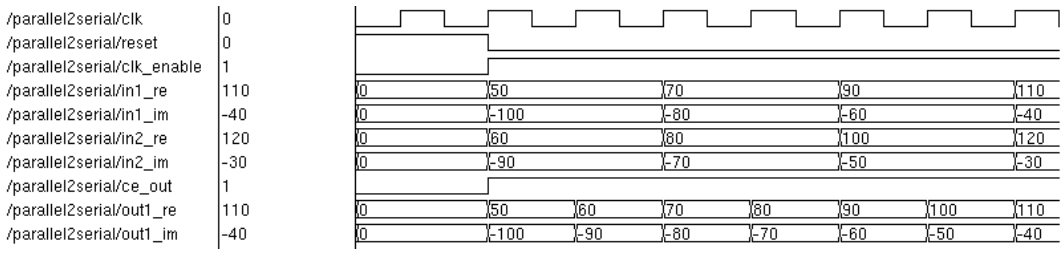


Figure 3.5: Timing diagram of the parallel-to-serial converter

Summary

This section reviews the issues, which, in author's opinion, facilitate understanding of this study. They concern the material in Sections 3.3 and 3.5 and cannot be regarded as research results.

Conclusions

- To prevent that the wordlengths of signal samples increase too much, bit truncation must be done after each multiplication.
- The rounding mode used for bit truncation affects the number of used *LEs* (see Section 3.3) and the error of elementary spectrum calculation (see Section 3.5)
- The kind of logic combination schemes (LUT or $ALUT$) in an *Altera FPGA* affects the number of used *LEs* – in case of $ALUT$ a smaller number of *LEs* is needed (on average $\approx 25\%$).

3.3 Implementation of the generalized Jacobi rotator

This section deals with two possible ways – traditional and using *CORDIC* algorithm – how to implement into *FPGA* chips the complex Jacobi rotator represented by the complex rotation matrix (1.3).

3.3.1 Traditional implementation using multipliers and adders [P9]-[P12]

For implementation of a single complex rotation 16 multipliers and 12 adders are needed (see Figure 3.6), if separately calculated products of sines and cosines (3.3) are being used. Given the fact that, in real systems, the real and the imaginary part of a complex signal are processed separately, the multiplication of a complex signal by the complex rotation matrix $\mathbf{T}_U(\phi,\ \psi,\ \gamma, 424)$ can be rewritten as follows, split into real and imaginary components:

$$\begin{cases} y_{1Re} = x_{1Re} \cdot ccg - x_{1Im} \cdot csg + x_{2Re} \cdot scp + x_{2Im} \cdot ssp \\ y_{2Re} = x_{2Re} \cdot ccg + x_{2Im} \cdot csg - x_{1Re} \cdot scp + x_{1Im} \cdot ssp \\ y_{1Im} = x_{1Im} \cdot ccg + x_{1Re} \cdot csg + x_{2Im} \cdot scp - x_{2Re} \cdot ssp \\ y_{2Im} = x_{2Im} \cdot ccg - x_{2Re} \cdot csg - x_{1Im} \cdot scp - x_{1Re} \cdot ssp \end{cases}, \tag{3.2}$$

where

$$\begin{aligned} ccg &= \cos(\phi) \cdot \cos(\gamma), \quad csg = \cos(\phi) \cdot \sin(\gamma), \\ scp &= \sin(\phi) \cdot \cos(\psi), \quad ssp = \sin(\phi) \cdot \sin(\psi). \end{aligned} \tag{3.3}$$

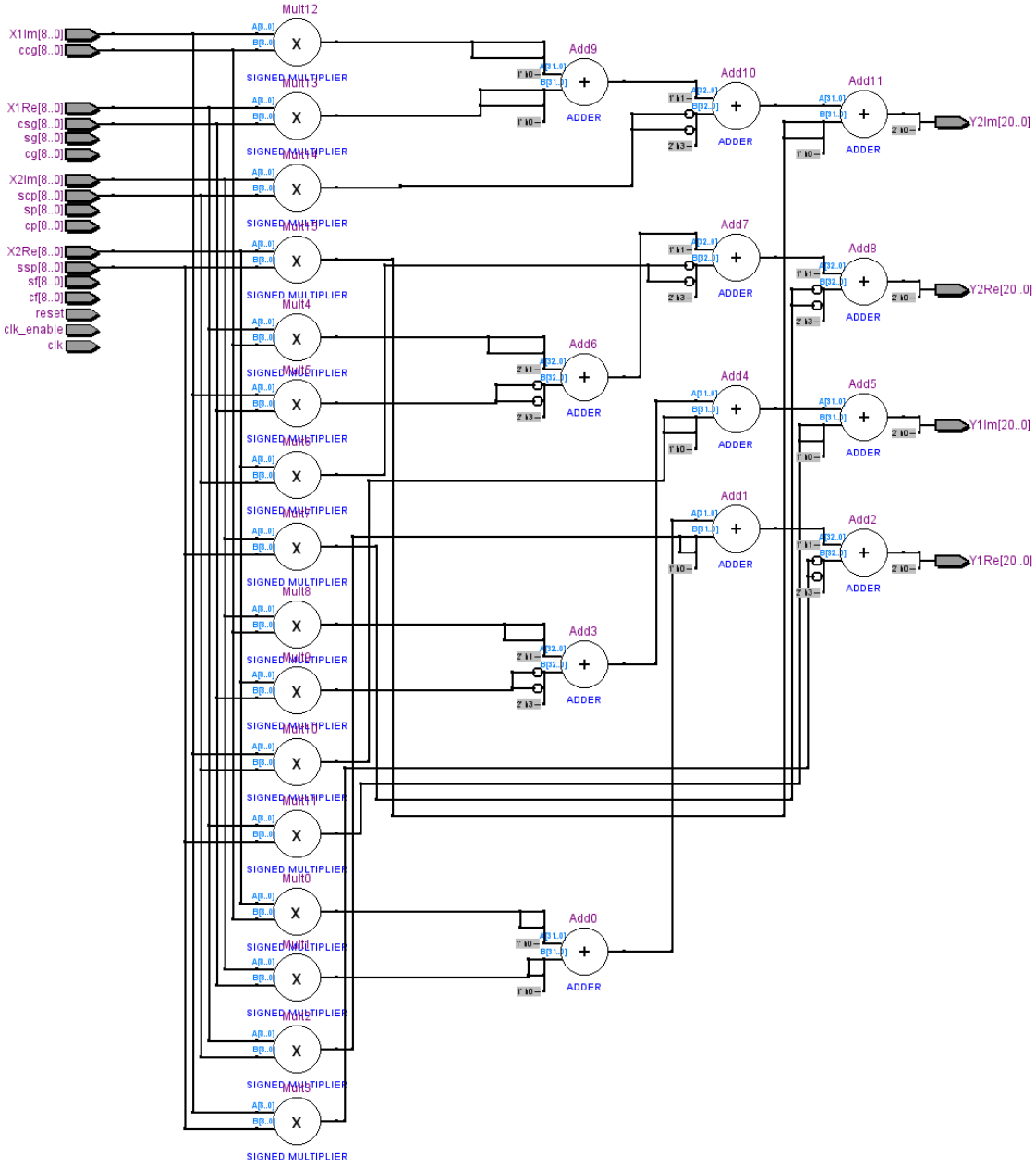


Figure 3.6: *RTL* description of the Jacobi rotator (no bit truncation)

Figure 3.7 shows, for the sake of simplicity, only the calculation of the Y_{Re} component using *fix* rounding.

When implementing algorithms into real devices, it is important to estimate the calculation time – the time required for a signal to pass from the input to the output. In case of *FPGA*, the number of used logic elements has to be also estimated. Table 7 shows the signal delay time and the number of used logic elements in dependence of the rounding mode and the wordlength of signal samples. In the table, w_{in} denotes the *WL* of input signal, w_{mult} – the *WL* of intermediate result after multiplication (if no bit truncation is

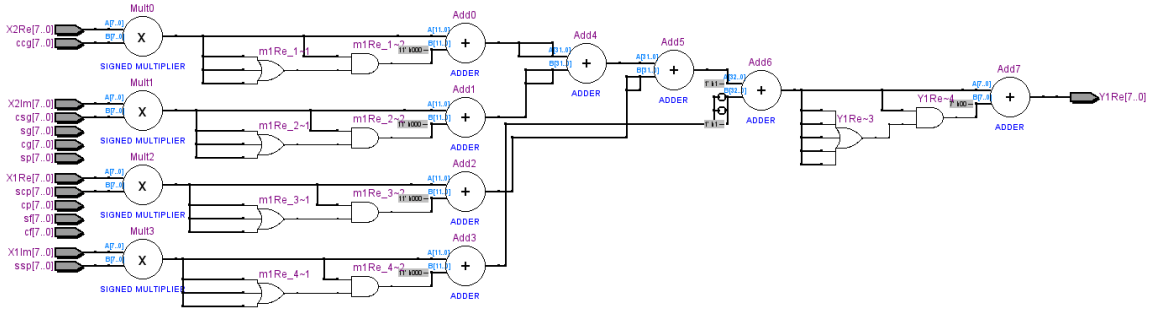


Figure 3.7: *RTL* description of the Y_{Re} component calculation, using *fix* rounding ($w_{in} = 8$, $w_{mult} = 12$, $w_{out} = 8$)

done, $w_{mult} = 2 \cdot w_{in}$), w_{out} – the *WL* of output signal. Note that changing the rounding mode and/or the *WL* of signals affects also the extent of *FPA* error (*FPA* errors are discussed in detail in Section 3.5).

Since parametrical angle-based transforms consist of several *REs* (see Section 2.3), it is important to keep *FPGA* resource consumption of each *RE* as low as possible. In view of that, *floor* rounding is accepted as optimal rounding mode for bit truncation (see Table 7). It is also the fastest of hardware rounding modes.

Table 7: Comparison of resource consuming, *TPD* and the number of *LEs* for some wordlengths and rounding modes, *EP2C35F672C6* (taken from [P12], complemented)

	Word lengths			**worst-case tpd**	**LEs**
	w_{in}	w_{mult}	w_{out}		
Fix	8	8	8	26.023 ns	334
	8	16	8	24.949 ns	251
	8	16	16	22.782 ns	192
Floor	8	8	8	20.551 ns	108
	8	16	8	22.494 ns	192
	8	16	16	22.782 ns	192
Round	8	8	8	26.483 ns	334
	8	16	8	25.799 ns	251
	8	16	16	22.782 ns	192
Nearest	8	8	8	22.762 ns	268
	8	16	8	23.188 ns	224
	8	16	16	21.577 ns	192

3.3.2 Implementation using *CORDIC* algorithm

For the time being, implementation of complex rotation using *CORDIC* algorithm is described only theoretically [45]. Therefore it is not possible to perform a precise comparison of required logic elements between this and the traditional way of implementation (see

Section 3.3.1). The load of programmable logic can be estimated from the real *CORDIC*-based implementation [P4]-[P7], [P9], because a single complex rotation can be performed as four real rotations [45].

The *CORDIC* algorithm is based on simplifying the elementary rotation expressions in a more convenient form for processing [46]. By transforming the elementary rotation expressions, the following is obtained:

$$\begin{aligned} x_{i+1} &= K_i(x_i - y_i \cdot d_i \cdot 2^{-i}) \\ y_{i+1} &= K_i(y_i + x_i \cdot d_i \cdot 2^{-i}) \end{aligned}, \tag{3.4}$$

where $K_i = \cos(\tan^{-1} 2^{-i}) = \frac{1}{\sqrt{1+2^{-2i}}}$,
$d_i = \pm 1$ (determines the rotation direction of the vector).

The rotation angle of the vector depends on the iteration index:

$$\phi_i = \tan^{-1}(2^{-i}),\ i = \{0..k\} \tag{3.5}$$

There are two approaches to implement the *CORDIC* algorithm in *FPGA*:

1. Unrolled – The elements of all *CORDIC* iterations (determination of the rotation direction, bit shifting, addition) are physically located on the *FPGA*. By varying the number of iterations (to obtain different precision), the number of used *LEs* varies considerably. The *CORDIC* algorithm implemented in such a way is easier to manage, but the time required to obtain the result depends on the number of iterations and the operation speed of *FPGA*.

2. Iterative – There are only the elements of one *CORDIC* iteration on the *FPGA*. On each clock pulse the results obtained in previous clock cycle are sent to the same *CORDIC* elements. If the *CORDIC* algorithm is implemented in such a way, even though several clock cycles are needed to obtain the result, the duration of each iteration depends only on the operation speed of *FPGA*. The number of iterations practically has no impact on the number of *FPGA LEs* used.

Table 8: Parameters of *FPGA* implementation of the *CORDIC* algorithm [*EP2C35F672C6*]

		Number of iterations					
		3	5	7	9	11	13
LE	unrolled	44	137	258	372	479	579
	iterative	234	273	274	299	302	304
min *Ts* (ns)	unrolled	13.4	22.9	31.5	42	49.3	62
	iterative	8	7.4	9	7.4	7.3	7.4
max angular error (%)		7.8	2	0.5	0.12	0.03	0.008

Table 8 shows the quantities concerning implementation of the *CORDIC* algorithm in *FPGA*. Depending on the selected architecture (unrolled or iterative) and the number of *CORDIC* iterations, the number of used *LEs* and also the maximum clock pulse frequency vary. In case of unrolled architecture, after time *Ts* shown in the table, the rotated vector is obtained, but in case of iterative architecture – the result of one iteration. By multiplying *Ts* by the respective number of iterations, the time can be calculated which is required to obtain the result in the iterative architecture of *CORDIC*. Table shows also the maximum possible angle error for *CORDIC* algorithm, expressed in percentages within the *CORDIC* operation range $[-90°, 90°]$.

It is not the most efficient approach to use only the *CORDIC* algorithm for rotation-angle-based transforms, because, besides *LEs*, *FPGAs* contain also *DSP* elements [42], which can perform comparatively fast *FPA* multiplication operations ($\sim$ 20 ns on *EP2C35F672C6*). When implementing transforms in pipeline architecture, the most efficient resource utilization can be achieved, using, at the first stages (see Section 2.3), the algorithms using *DSP* elements (because of their high operation speed), at the remaining stages – combinations of *CORDIC* (no need for *DSP* elements, the number of which is very limited on *FPGA*) and algorithms using *DSP* elements.

Summary on papers [P5], [P6], [P9]-[P12]

Obtained results

- Analysis of two ways of implementation of the complex elementary rotation in FPGA:
  - traditional, using multipliers and adders,
  - using *CORDIC*.

Conclusions

- When performing the rotation using *CORDIC* algorithm, no *DSP* elements are used.
- To perform the complex rotation with three parameters (variable angles ϕ, ψ un γ), 20 multiplications and 12 additions are required.
- Any way of implementation of the *CORDIC* algorithm (with five or more iterations – specifically, using the chip *EP2C35F672C6*) is slower than the vector rotation using *DSP* multipliers. It is characteristic for all *Altera* chips with integrated multipliers.
- An acceptable rotation precision can be achieved with at least nine *CORDIC* iterations, that is, with 9 iterations the maximum angle error is 0.12%, but, for example, with 13 iterations – 0.008%.

- The number of *DSP* elements on *FPGA* is limited – when no more *DSP* elements are available, multiplication is performed using *LEs*. That increases *LE* consumption considerably – ≈ 110 *LEs* per one 9-bit *DSP* element.

- When the traditional implementation of *RE* is used at the first stages of tree-like algorithms, the highest operation speed is achieved. But if the *CORDIC* algorithm is used, the number of required *DSP* elements decreases. It is recommended that when developing devices with the highest possible operation speed, at the first stage (for signal reconstruction – at the last stage) the traditional algorithm should be used and at the remaining stages – *CORDIC*.

- Using the traditional algorithm, the chosen rounding mode for bit truncation affects both the number of used *LEs* and the operation speed – *floor* rounding does not require additional *LEs* and in this case the algorithm is executed with the highest operation speed. For example, comparing floor and round rounding – if the *WLs* of signal samples are ($w_{in} = 8$, $w_{mult} = 8$, $w_{out} = 8$), then *floor* needs three times less *LEs* and the result is obtained in 20% less time (*EP2C35F672C6*).

- The complex rotation using *CORDIC* is not yet implemented in *EGURIT*, still it is obvious that the complex rotation can be performed using four real rotations (that is, real *CORDICs*). It means that the results, obtained in this research, concerning resource consumption of the real *CORDIC* can be applied also to the *CORDIC*-based complex rotations.

3.4 Simplifications of the elementary generalized complex rotation matrix

The research has shown that it is not always necessary to implement the whole 3-parameter complex rotation matrix (*EGURM*). There are a number of transforms for which a simplification of the matrix, where some parameter is substituted by a constant, is sufficient. For example, if $\psi = \frac{\pi}{2}$ and $\gamma = 0$, then (1.6) reduces to:

$$\mathbf{T}(\phi, \psi = \frac{\pi}{2}, \gamma = 0,424) = \begin{bmatrix} \cos(\phi) & -i \cdot \sin(\phi) \\ -i \cdot \sin(\phi) & \cos(\phi) \end{bmatrix} \tag{3.6}$$

By splitting the product of a complex vector $\begin{bmatrix} y_1 \\ y_2 \end{bmatrix}$ and the complex matrix (3.6) into real and imaginary parts, the following equations are obtained:

$$\begin{cases} y_{1Re} = x_{1Re} \cdot \cos(\phi) + x_{2Im} \cdot \sin(\phi) \\ y_{2Re} = x_{1Im} \cdot \sin(\phi) + x_{2Re} \cdot \cos(\phi) \\ y_{1Im} = x_{1Im} \cdot \cos(\phi) - x_{2Re} \cdot \sin(\phi) \\ y_{2Im} = -x_{1Re} \cdot \sin(\phi) + x_{2Im} \cdot \cos(\phi) \end{cases} \tag{3.7}$$

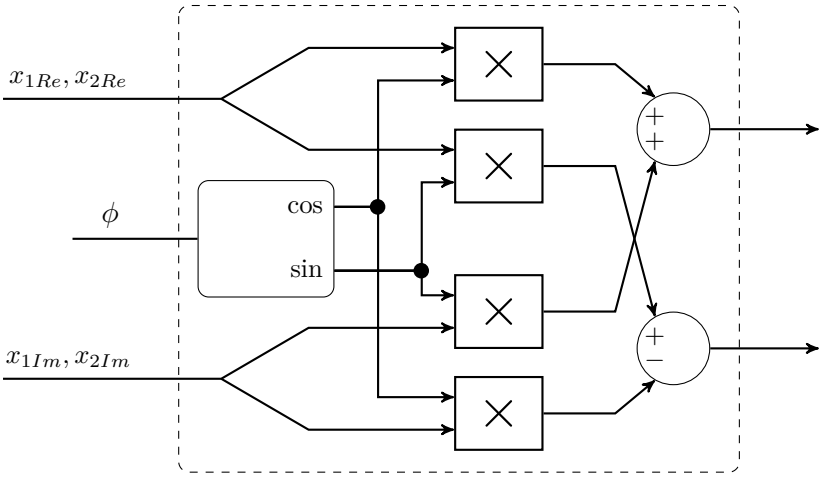


Figure 3.8: Simplified block diagram of serial *EGURM*-rotator for (ϕ, $\psi = \frac{\pi}{2}$, $\gamma = 0$, ID=424) [P9]

These expressions are used for implementation of rotation element (*RE*) into *FPGA*. Figure 3.8 shows a simplified block diagram of implementation of the equations (3.7). For each simplification and structure of the rotation matrix a separate *VHDL* code must be created, and, therefore, their number can reach several thousands [P12]. This task would be very time-consuming, so an automated system has been created (see Section 3.6) to check all structures and simplifications of the rotation matrix and generate corresponding *VHDL* code.

Table 9: The number of operations for some sets of angles and rotators without using memory [P11]

ϕ	*var*	*var*	*var*	$\pi/2$	*var*	*var*	$\pi/2$
ψ	*var*	*var*	$\pi/4$	$\pi/4$	0	$\pi/2$	$\pi/2$
γ	*var*	0	$\pi/4$	*var*	0	$\pi/2$	any
M	20	14	8	0	8	0	0
A	12	8	4	4	4	4	0
MC	0	0	4	4	0	0	0

The using of simplifications reduces the number of necessary arithmetic operators. Table 9 shows the number of multiplications and additions, including the mutual products of sines and cosines (3.3), needed for the complex rotation.

Summary on papers [P9], [P11] un [P12]

Obtained results

- A symbolic math algorithm, built in *EGURIT* [P12], which allows to find , for a given set of angles, all the possible *EGURM* simplifications (*faces*) and expressions for the corresponding elementary spectrum.

Conclusions

- By fixing any of the parameters (ϕ, ψ or γ) to a constant value, it is possible to reduce essentially the number of multiplications and additions needed for vector rotation. But it also means that the number of possible variations of RE increases. For example, by setting six different values (*variable*, 0, $\pi/6$, $\pi/4$, $\pi/3$, $\pi/2$) for each parameter and applying it for all the possible RE structures (*shapes*), it can be calculated that the total number of different RE modifications is 6880 [P12].

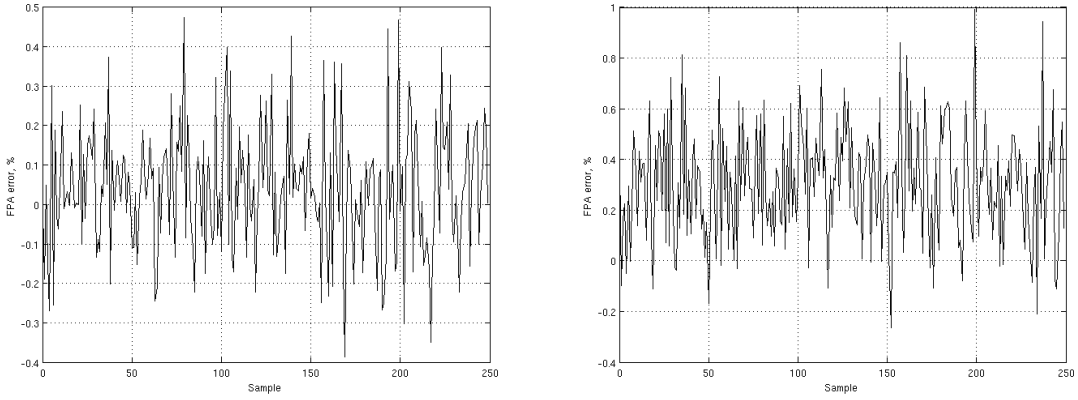
- Choosing constant values to be, for example, 0, $\frac{\pi}{4}$, $\frac{\pi}{2}$, some (up to 7) variations of standardized expressions can be obtained. They differ by the number of M, A and MC.

- To manage the immense amount of $EGURM$ simplifications and corresponding elementary spectra, it is necessary to automate the process of generating these expressions.

3.5 Fixed-point error

This section discusses a fixed-point error at the output of RE element, depending on the wordlength (WL) of input signal samples and intermediate signal samples (inside RE).

One of the sources of fixed-point errors is the truncating of signal sample WL. The scaling of WLs must be done when implementing algorithms in fixed-point devices. Depending on the WL of input, intermediate and output signals, the error for obtaining the output signal (spectrum of two samples) varies. Table 7 in Section 3.3.1 shows how the number of used logic elements vary for different signal WLs.



(a) $w_{in} = 8$, $w_{mult} = 12$, $w_{out} = 16$

(b) $w_{in} = 8$, $w_{mult} = 12$, $w_{out} = 8$

Figure 3.9: Fixed-point errors for a 250-samples long noise-like signal

Figure 3.9 shows fixed-point errors for each sample of a 250-samples long noise-like signal, when the number of bits in intermediate results after multiplication increases by 4 bits. The errors are expressed in percentages of the fixed-point dynamic range and calculated according to the following formula:

$$\epsilon_{\%} = \epsilon_k \cdot 100\% = \frac{\widehat{\mathbf{v}} - \mathbf{v}}{r} \cdot 100\%, \tag{3.8}$$

where $\widehat{\mathbf{v}}$ – signal with a fixed-point error, $\mathbf{v}$ – signal with no fixed-point error (obtained using *MatLab* floating point), r – fixed-point range.

An *FPA* error distribution histogram, obtained for a 100000-samples long noise-like input signal, is shown in Figure 3.10.

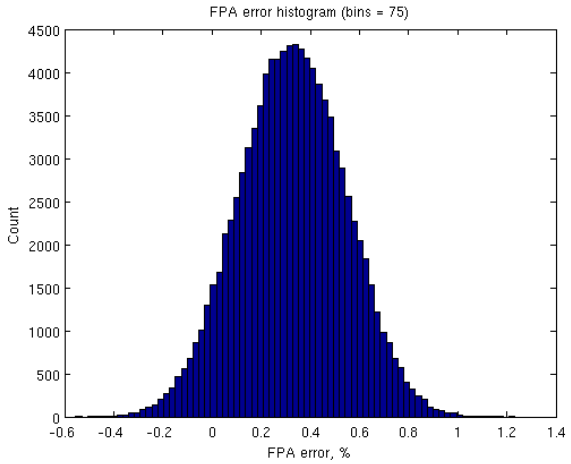


Figure 3.10: Distribution of *FPA* errors ($w_{in} = 8$, $w_{mult} = 8$, $w_{out} = 8$, *floor* rounding)

For comparisons it is much more convenient to use the mean square error, which is calculated as follows:

$$\epsilon_{MSE} = \sqrt{\frac{1}{N}\sum_{k=1}^{N} {\epsilon_k}^2} \tag{3.9}$$

Also the chosen rounding mode (see Section 3.2.1) affects the extent of fixed-point error. Table 10 shows fixed-point *MSE* and Table 11 – maximal *FPA* error modulus for all

Table 10: Fixed-point *MSE* ϵ_{MSE}

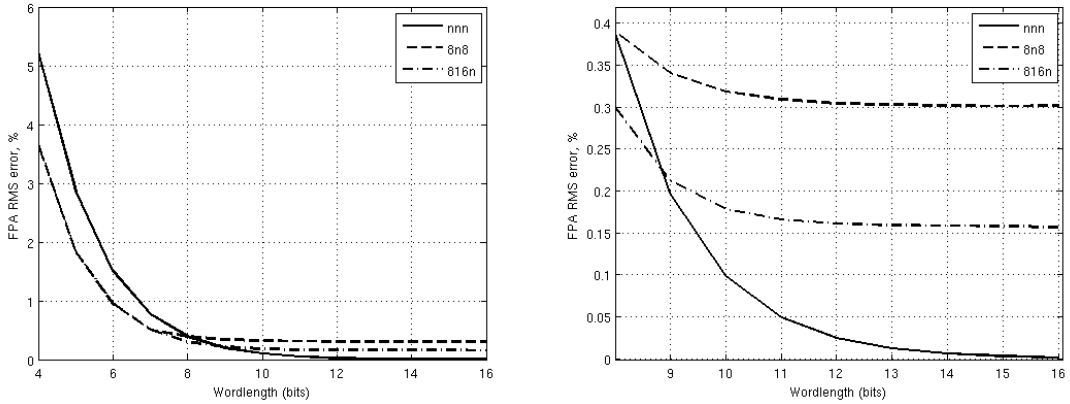
Signal properties			Rounding Type			
w_{in}	w_{mult}	w_{out}	Floor	Fix	Round	Nearest
8	8	8	0.39%	0.40%	0.21%	0.22%
8	16	8	0.30%	0.32%	0.19%	0.19%
8	16	16	0.16%	0.16%	0.15%	0.15%

Table 11: Maximal fixed-point error modulus $max(abs(\epsilon_{\%}))$

Signal properties			Rounding Type			
w_{in}	w_{mult}	w_{out}	Floor	Fix	Round	Nearest
8	8	8	1.29%	1.38%	0.89%	0.96%
8	16	8	0.95%	0.96%	0.82%	0.83%
8	16	16	0.65%	0.68%	0.65%	0.66%

possible rounding modes and different signal *WLs*. To calculate the mean square error and the modulus of maximal error, 100000-samples long data sequences are used, generated using the *MatLab* built-in noise generator with *mrg32k3a* algorithm.

Figure 3.11 shows how *FPA* errors change by varying *WLs* in different ways. The



(a) $w = \{4..16\}$ (b) $w = \{8..16\}$

Figure 3.11: Fixed-point *MSE*, for different *WL* (*floor* rounding)

notation "*nnn*" ("$w_{in}w_{mult}w_{out}$") means that *WLs* of input signal (w_{in}), signal after multiplication (w_{mult}) and output signal (w_{out}) are varied equally. "8n8" means that *WL* of input and output signals are fixed to 8 bits, but the *WL* of signal after multiplication is varied. "816n" – the *WL* of input signal is 8 bits, after multiplication – 16 bits, and the *WL* of output signal is varied.

Summary (the results have not been published)

Although the results (graphics, tables, formulas) have not been published, a tool for error estimation is included in *EGURIT* ([P12]).

Obtained results

- A tool for *RE FPA* error estimation.

Conclusions

- The developed tool allows to calculate mean square errors, arising in performing an *EGU*-rotation, for different:
  - wordlengths of input signals, intermediate signals (after multiplication) and output signals,
  - rounding modes.
- The developed tool should (is planned) be improved to allow to determine automatically optimal wordlengths and rounding modes for a given error.

3.6 Automation of implementation of *EGU*-rotator [P11], [P12]

As mentioned in Section 1.2.1, the number of rotation matrix structures (*shapes*) is 32, when $\phi, \psi, \gamma \in [0, 90°]$. Considering all the possible modifications of *EGU*-rotation matrix, the total number of different *EGU*-rotator realizations can reach even several thousand. Thus it is evident that the manual management of implementation of *EGU*-rotator cannot be real. This confirms the necessity for the implementation automation.

The following software is used for implementation automation: *MatLab/Simulink*, *Quartus II* and *ModelSim*. The main tasks for each software are as follows:

- *MatLab* – Automation control, *EGU*-rotation matrix *shape* unitarity test, equation simplification and separation into real and imaginary parts (by using *Symbolic Math Toolbox(SMT)*), results calculation in floating point (double precision) and fixed point arithmetics.
- *Simulink* – System simulation in *FPA*, *VHDL* code generation using *HDL coder*.
- *ModelSim* – *VHDL* code simulation.
- *Quartus II* – *VHDL* code compilation for chosen *FPGA*, *FPGA* programming.

MatLab/Simulink is the default standard for complex *DSP* tasks. *Quartus II* software is chosen mainly because of practical experience with it and availability of adequate *FPGA* design kits (which are purchased while working on several projects). *Mentor Graphics ModelSim* is *Altera's* recommended simulation tool. Command line scripting is available for both *Quartus II* and *ModelSim* software. This allows to perform *VHDL* code simulation and compilation directly from *MatLab*. We can split the whole automation process into several activities (more detailed described in Section 3.6.1):

- Using Symbolic Math Toolbox,
- *FPA WL* choice,

- *MatLab* function generation,
- *HDL* code generation,
- Testing.

3.6.1 Design automation steps

The main design flow is depicted in Figure 3.12, where "*EGURM shape*" stands for *EGURM* shape selection, "*EGURM face*" – modification choice, "*Text proc.*" – symbolic calculation and simplification of basic expressions, "*FPA fi()*" – *MatLab's FPA* object construction for obtained equations.

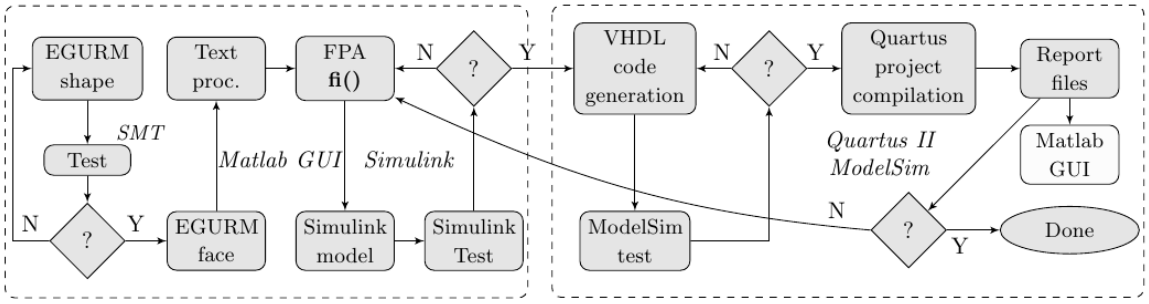


Figure 3.12: Simplified design automation flow chart [P12]

Using symbolic math toolbox The first thing in this activity is the orthogonality (unitarity) test for all possible structures (in developed tools (see Figure 3.13) indicated as "*shapes*") of complex rotation matrix (1.3). Orthogonal structures then are saved in database. Next, the simplification ("*Face*", see Figure 3.13) is chosen, for instance, $(\phi, \psi = \frac{\pi}{4}, \gamma)$. With *MatLab* function **latex()** the LaTeXform of equations is obtained.

Then, similar to (3.7), complex equations are separated into real and imaginary equations. Finally, obtained equations need to be split into basic expressions in such a way, that each expression consists of one multiplication or one addition. For example, by splitting (3.7), $y_{1\,Re}$ is obtained as follows:

$$\begin{aligned} m_{1\,Re\,1} &= x_{1\,Re} \cdot \cos(\phi) \\ m_{1\,Re\,2} &= x_{2\,Im} \cdot \sin(\phi) \\ y_{1\,Re} &= m_{1\,Re\,1} + m_{1\,Re\,2} \end{aligned} \tag{3.10}$$

***FPA WL* choice** In this activity *WLs* for intermediate results are obtained. This can be done manually or using special algorithm. When *MatLab's fi()* object is used we should enter both the total *WL* of signal and the fraction *WL*. We should keep in mind that vector rotation is done by algorithm. Because of that, without input signal vector length scaling, output signal can exceed $Q1.x$ *FPA* format limits (when input signal is in $Q1.x$ *FPA* format). For example, when the vector $[-1,\ -1]$ is rotated clockwise by 45° we obtain $[0,\ -\sqrt{2}]$. This is the reason why output signal is in $Q2.x$ *FPA* format, which gives range $[-2,\ \ 2-\frac{1}{2^x}]$. $Q3.x$ or $Q4.x$ *FPA* formats could be needed for intermediate results (depending on adders count).

The following principles are used for signal *WL* calculation algorithm:

- Chosen multiplier output signal *WLs* are set (w_{mult}), keeping $Q1.x$ *FPA* format.
- When signal is multiplied with a constant coefficient, the *WL* and the number of fraction bits stay unchanged,
- The adder output signal *WL* is increased in such a way, that the number of fraction bits stays unchanged.

MatLab code generation From symbolic equations and *FPA WLs* a *MatLab* function for *HDL* code generation must be obtained. The using of *Embedded MatLab Function* (*EMLF*) in *Simulink* is more preferable than building corresponding *Simulink* block diagram using ***add_block()*** function. A *MatLab* function can be used as *EMLF* when **%#eml** compilation directive is added in function kernel.

HDL code generation The *EMLF* with the reference to the external *MatLab* function is inserted into *Simulink* model. All outputs (*X1Re*, *X1Im*, *cf*, etc) must be added and the $Q1.x$ *FPA WL* of input signals must be specified. Symbolic expressions are constructed so that they contain only real multiplication and addition operations, and therefore, input parameters include also $\sin(\phi)$, $\cos(\phi)$, $\sin(\psi)$, etc. In order to have as few as possible multipliers in the rotator module, the necessary *sin/cos* products, for example, $ccg = \cos(\phi) \cdot \cos(\gamma)$, are also given as input parameters. The corresponding *HDL* code is generated by calling *MatLab* function **makehdl()**. When correspondingly configured, it generates also *Quartus II* and *ModelSim* scripts (see also Section 3.6.2).

Testing Three different tests are used:

- *MatLab* floating point test,
- *Simulink* with *FPA* test,
- *HDL* code *ModelSim* test.

Input signals for all tests are generated using *MatLab* random number generator with *mrg32k3a* algorithm (uniform distribution). This algorithm has multiple stream support and approximate stream period is 2^{127} samples. Generated input signal streams are formed – as *MatLab* vectors for floating point test, as a structure in *MatLab* workspace for *Simulink* test, and as a *.do file for *ModelSim* test. Floating point and *Simulink* test results are used for *MSE* calculations. *ModelSim* test results are compared to *Simulink* test results to make sure that *VHDL* code works properly.

3.6.2 Automation tools

The developed automation tools (MatLab GUIs) and their main functions are listed below.

Rotation Matrix Viewer (see Figure 3.13) ($\sim$ 750 *Matlab* lines)

- *EGURM shape* choice,
- *EGURM face* choice,
- *Face* parameter combination choice,
- LaTeXrepresentation of symbolic equations.

This tool is used to choose an *EGURM* shape to be implemented in *FPGA*. It uses a database of all valid shapes, created beforehand. This tool allows also to obtain a simplifications (*face*), if needed, for a chosen *shape*.

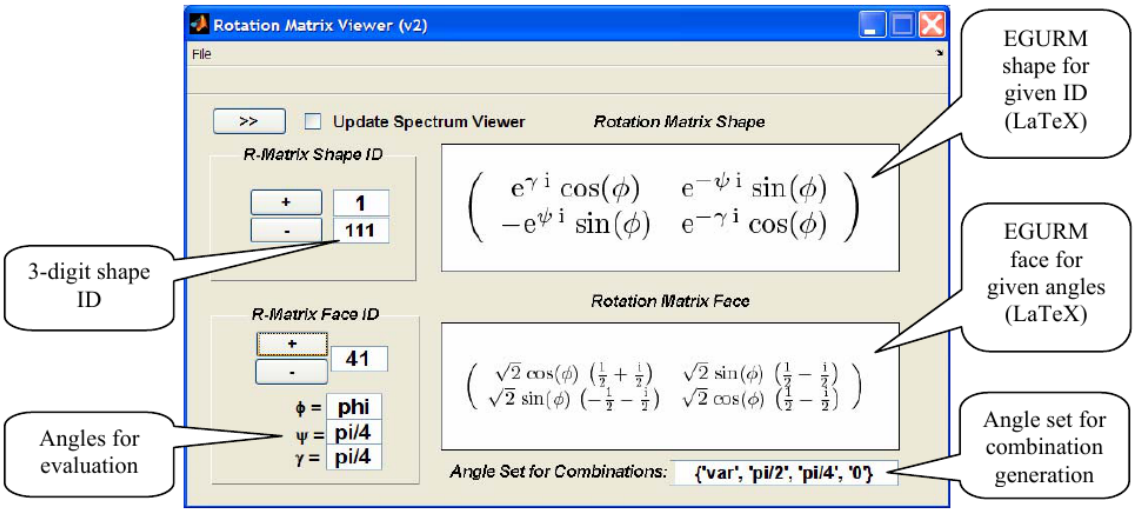


Figure 3.13: Rotation Matrix Viewer *Graphical User Intarface GUI* [P12]

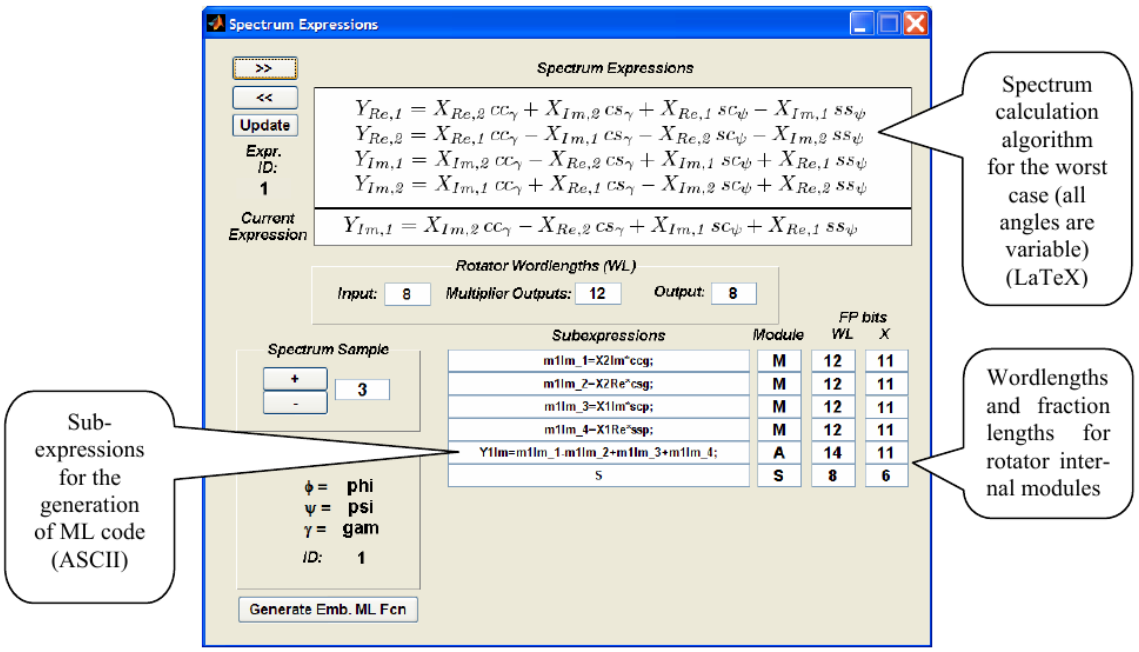


Figure 3.14: *Spectrum Expressions GUI* [P12]

Spectrum Expressions (see Figure 3.14) (∼ 2100 *MatLab* lines)

- LaTeXrepresentation of spectrum calculation expressions,
- Spectrum coefficient choice and *ASCII* expressions for calculation,
- Automatic/manual setting of signal *WL*,
- Generation of *Embedded MatLab function*.

This tool is used to set *FPA WLs* and generate a *MatLab* function for *HDL* code generation. This function is used in the *Simulink* model shown in Figure 3.15.

HDLCoderGUI (see Figure 3.16) (∼ 600 MatLab lines)

- Floating point and *Simulink* test,
- Configuration of *Simulink HDL coder*,
- Generation of *VHDL* code and *Quartus II* and *ModelSim* scripts,
- Creation of *Quartus II* project folders,
- *ModelSim* test and comparison of *Simulink* and *ModelSim* test results,

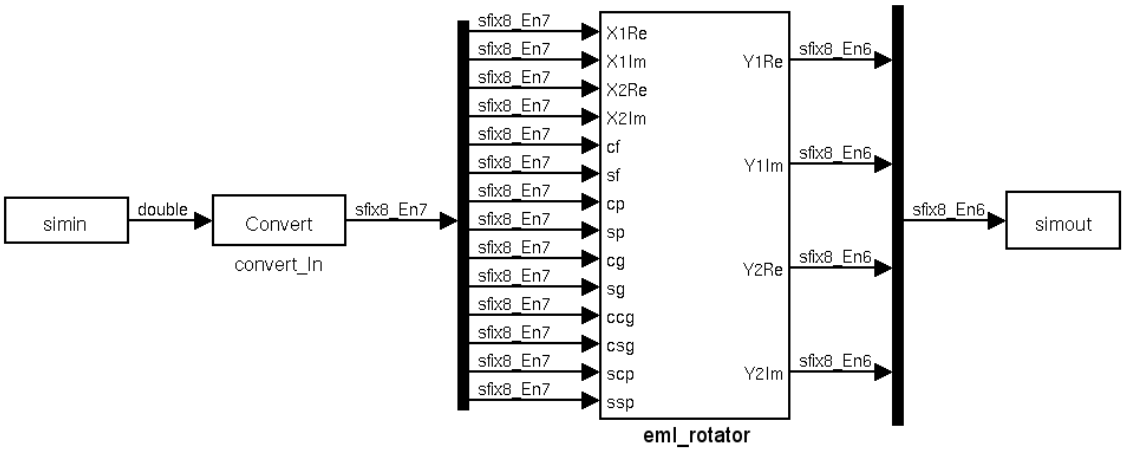


Figure 3.15: *Simulink* model for *EGU*-rotator *VHDL* code generation [P12]

- *Quartus II* compilation,
- Display results.

After *HDL coder* configuration the following files are generated automatically: *Quartus* scripting file *projNosaukums_quartus.tcl* and *ModelSim* scripting file *projNosaukums_compile.do*. These two files are used for *Quartus* and *ModelSim* project creation and compilation on the command line. When the ModelSim project is compiled, simulation is performed. Simulation results are saved in **.lst* file.

The capability to perform *Quartus* and *ModelSim* on the command line allows to execute *Quartus* and *ModelSim* software from *MatLab*, by using *MatLab* function **system()**. Each *EGU*-rotator project is compiled in a separate folder, which name contains information on time and date.

Some time resources needed for the *EGURIT* compilation and simulation can be found in Table 12. When performing *EGU*-rotator automation, we need to pay attention to

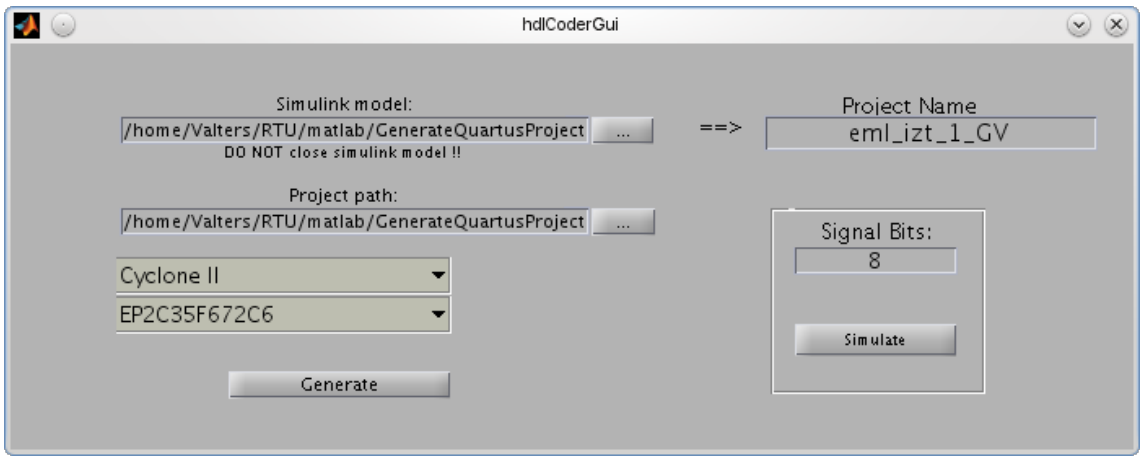


Figure 3.16: *HDLcoderGUI Graphical User Interface*

Table 12: Compilation/simulation time (Linux, Dell Latitude D 820) [P12]

Action	*Approx. time (s)*
Simulink simulation (2500 points) with EMLF code generation	4.5
Simulink simulation (2500 points)	0.7
VHDL code and script file generation from Simulink model	2
Folder creation	0.003
ModelSim simulation	2
Quartus project compilation (time depends on the used device and algorithm)	38 - Cyclone C12, FA* 545 -StratixIV E820, FA 28 - Cyclone II C35, SA* 45 - Cyclone II C35, FA

*FA – the full algorithm, SA –the simplest algorithm

Quartus project compilation time as it heavily depends on the parameters of chosen *FPGA*. It is not practical to choose very powerful *FPGAs* (*Stratix IV E820*, for example) for *FPA* error estimation.

Summary on papers [P11] and [P12]

Obtained results

- Situation in automation tool design, that is oriented to *VHDL* codes synthesis related to Jacobi rotations, has been analyzed.
- The automation tool *EGURIT* for *EGU*-rotator *VHDL* code synthesis in *Altera's FPGA* has been developed. The following software is used for *EGURIT*:
  - *MatLab/Simulink*,
  - *Altera Quartus II*,
  - *Mentor Graphics ModelSim*.

Conclusions

- In available literature no information on tools similar to *EGURIT* is found.
- Analysis of available literature shows that *MatLab/Simulink* software is the most common environment for automated *DSP* system development.
- Analysis of available literature shows that, most commonly, *Altera's* and *Xilinx's FPGA* are used for *DSP*, and *VHDL* codes are synthesized for *FPGAs* manufactured by these companies.

- *Quartus II* and *ModelSim* command line scripting allows development of effective automation tools.

- The developed automation tool allows synthesizing of *VHDL* codes and their implementation in *FPGA* for any of several thousand of *EGU*-rotator *shapes* and *faces* (see summary of Section 3.4 or [P12]). The consumption of *FPGA* resources (the number of *LEs* and *DSP* elements, timing parameters) can be determined using this tool, as well.

- In a relatively short time *EGURIT* allows:

  - To obtain parameters for a huge number of different *RE shapes* and *faces*,
  - To estimate implementation options of rotation-angle-based transforms for $N > 2$ (to estimate potential *FPGA* resources – *LEs*, *DSP* elements, timing parameters).

- Time performance of the developed tool is determined by *Quartus II* project compilation time (from some seconds to several minutes). This, in its turn, mainly depends on *FPGA* chip complexity and computer performance. It is recommended to choose chips with a smaller number of *LEs* for *FPGA* resource estimation.

- The developed automation system can also be adapted for *FPGA* development software from other manufacturers, for instance, *Xilinx ISE*, *Synopsis*, etc. (generally speaking, for such development software which allows command line scripting).

4 Experimental Devices

4.1 *FPGA* implementation of experimental CRAIMOT function generator

[P1] and [P4] describe the details about **CRAIMOT** *BF* generator. We can implement a generator depicted in Figure 2.1 using *Quartus II* software (see Figure 4.1). The *Quartus II* graphical environment has not been used in further versions of the devices because it is more efficient to write directly in *VHDL* (all blocks of depicted block diagram are written in *VHDL* ($\approx$ 300 strings), but graphical environment is used only for block interconnection), and simulation in graphical environment is not supported starting from *Quartus II ver. 10.0*.

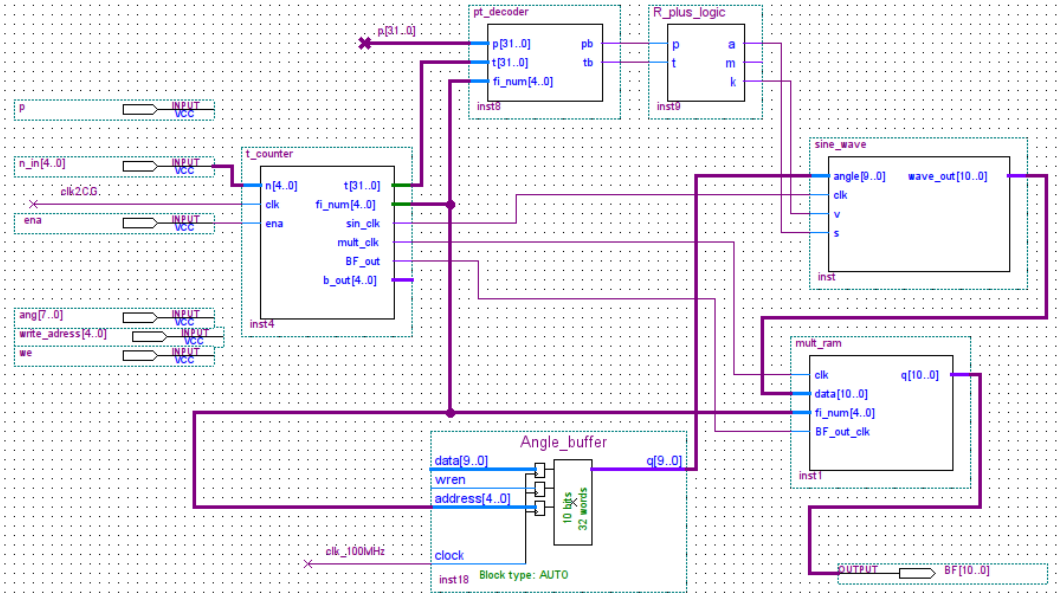


Figure 4.1: **CRAIMOT** *BF* generator implemented using *Quartus II 6.0* graphical environment

The parameters of the generator *FPGA* implementation are summarized in Table 13. Simulation timing diagrams of *BF* generator are shown in Figure 4.2. For demonstra-

Table 13: The list of parameters of *BF* generator with serial architecture [P1]

Parameter	Value
External clock used	$f_{clk} = 48$ MHz
Maximal number of angles	$n = 32$
Max. *BF* length (samples)	$N = 2^n = 4294967296$
Wordlength for *BF* values	$w = 10$ biti, $Q1.x$ *FPA*
BF sample duration time	$T_{s\,BF} = (3 \cdot n + 1)/(f_{clk}/4)$
BF repeating period	$T_{BF} = 1/(N \cdot T_s)$

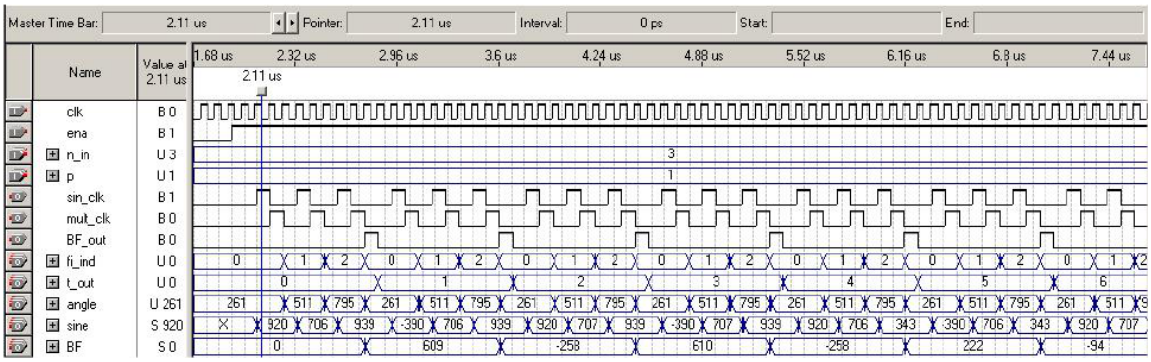


Figure 4.2: Simulation timing diagram for **CRAIMOT** *BF* generator (from *Quartus II 6.0*) [P1]

tion purposes, a digital signal is converted to analog signal and shown on the oscilloscope screen using the audio codec from the *FPGA* development kit. Figure 4.3 shows a comparison of the data imported from oscilloscope and the data generated by *MatLab*.

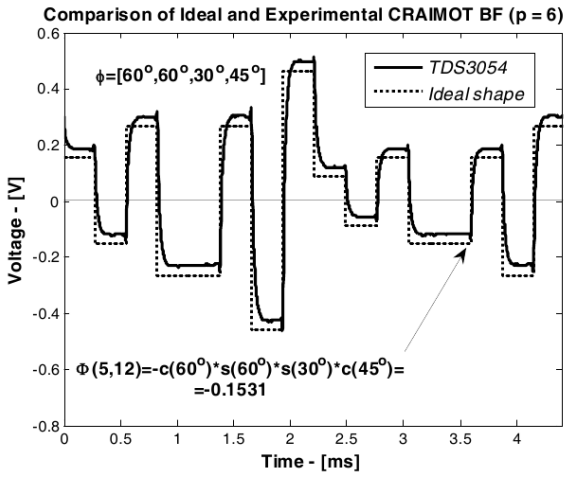


Figure 4.3: Shapes of ideal *BF* and experimental *BF* for $p = 6$ and $N = 16$ (the shape of captured *BF* is shifted up slightly for better comparison) [P1]

The **CRAIMOT** *BF* generator with serial architecture [P1] is one of the first from practically implemented experimental angle-based devices.

Table 14: Consumption of hardware resources for *sin/cos* block ([P4], improved)

	Logic elements (LE)	*9-bit Multipliers*	*RAM bits*
9-bit CORDIC	$n \cdot 372$	0	0
Linear Interpolation with 6 knots	$n \cdot 161$	$n \cdot 10$	0
RAM-based table	0	0	$n \cdot 2^{n_\alpha} \cdot w$

Table 15: The list of parameters of *BF* generator with parallel architecture [P4]

Parameter	*Value*
External clock used	$f_c = 100$ MHz
Maximal number of angles	$n = 6$
Max. *BF* length (samples)	$N = 2^n = 64$
Wordlength for *BF* values	$w = 10$ bits, *Q1.x FPA*
BF sample duration time (ns)	$T_{sBF} = 40 \cdot (n-1)$
BF repeating period	$T_{BF} = 1/(N \cdot T_s)$

Also the *BF* generator with parallel architecture [P4] was developed at the same time. The *sin/cos* value generation component is included in the both architectures. *BF* value error and the amount of used resources (see Table 14) depend on the kind of *sin/cos* generation (linear interpolation, internal *RAM*, *CORDIC*). *VHDL* codes have been continuously improved, due to growing professional competence. This is a reason why data in Table 14 differ from data provided in [P4]. In implementation of *BF* generator with parallel architecture an *FPGA* advantage can be exploited – parallel structured elements, which ensure higher performance in comparison with serial architecture (see Table 15). The number of *LEs* depends on the *BF* length, because for the generation of *BF* sample are necessary $n = \log_2(N)$ *sin/cos* blocks simultaneously (see Figure 4.4).

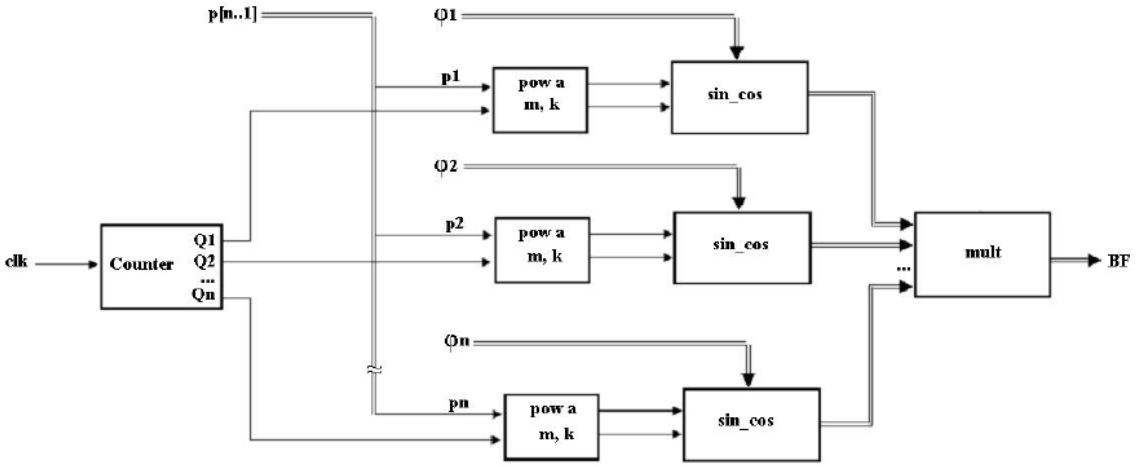


Figure 4.4: Simplified block diagram of **CRAIMOT** *BF* generator with parallel architecture [P4]

Summary on papers [P2] and [P5]

Obtained results

- Two versions of *FPGA*-based **CRAIMOT** *BF* generator.

- An expression for the calculation of *BF* minimal sample time (for implementation into *EP1C12* chip at 48 MHz clock).
- Estimation of consumed resources for three kinds of *sin/cos* value generation:
  - using *CORDIC* algorithm,
  - using optimized linear interpolation (*LININT*),
  - using *RAM* table.

Conclusions

- The serial architecture of generator allows forming of very long *BFs* ($N = 2^{32}$) for a relatively small number of allocated *LEs*.
- The minimal sample time changes linearly in dependence on the number of angles (n). E.g., if the number of angles is varied from 5 to 12, the minimal sample time changes from ~ 1 up to 3 microseconds.
- The length of *BF* does not impact the number of used *LEs* for *BF* generator with serial architecture.
- Performance of *BF* generator with serial architecture is more than sufficient for the synthesis of audio signals.
- The sampling frequency of implemented *BF* generator with parallel architecture [P4] is five times higher than that of *BF* generator with serial architecture [P1].
- The number of used *LEs* for implemented *BF* generator with parallel architecture depends linearly on the number of angles (or logarithmically, on the length of *BF*) and a *sin/cos* value generation method. E.g., for *CORDIC* algorithm, the number n of used *LEs* is equal to $n \cdot 372$.
- It is ascertained that the *sin/cos* component(s) (block(s)) consume(s) most of *FPGA* resources used for the generator.
- The *RAM* table is the most efficient for *sin/cos* value generation, taking into account further technology development and increasing of the number of *RAM* bits on *FPGAs*. *LEs* and *DSP* blocks are not used in this case.
- The algorithm of optimized linear interpolation is useful for *sin/cos* value generation only if there are unallocated multipliers on the chip.

4.2 *FPGA* implementation of experimental signal spectrum analyzer

4.2.1 Signal spectrum analyzer with serial architecture [P2]

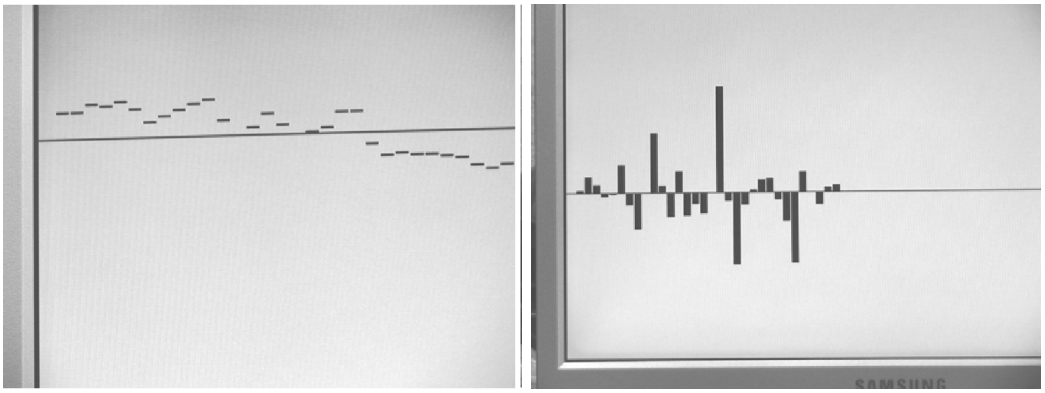
[P2] describes the details about implemented *CRAIMOT* spectrum analyzer that includes *BF* generator. Table 16 shows the list of parameters for the real-time spectrum analyzer based on BF generator.

Table 16: The list of parameters of spectrum analyzer [P2]

Parameter	Value
External clock used	$f_{clk} = 48$ MHz
Signal block length for: •real-time mode: •capture mode:	 $N = 32$, for $F_s = 44.1kHz$ $N = 64$, for $F_s = 8.2kHz$ $N = 1024$
Woedlength for signal samples and spectrum coefficients	$w = 10$ bits, $Q1.x$ *FPA*

A simple *VGA* video controller is added to the analyzer for demonstration of input signal and obtained **CRAIMOT** spectrum on *LCD*. Figure 4.5 shows respective screen snapshots.

This 1-D spectrum analyzer is the first of such a kind of experimental *FPGA*-based devices, and it has established the ground for further development of *DSP* devices based on more efficient algorithms (e.g., using *RE*).



(a) a) Piece of the input signal of spectrum analyzer

(b) **CRAIMOT** domain spectrum

Figure 4.5: LCD snapshots [P2]

4.2.2 *RE*-based signal spectrum analyzer-synthesizer [P5]

This spectrum analyzer performs the calculation of spectrum using parallel input data. A simplified spectrum calculation block diagram is shown in Figure 4.6. By "*CORDIC*

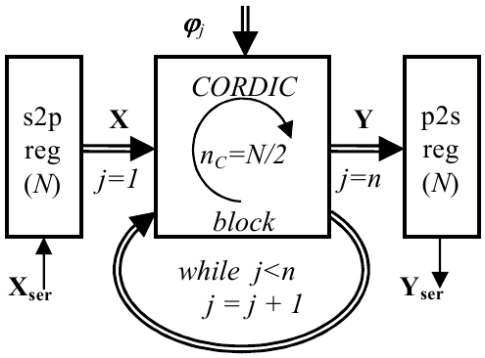


Figure 4.6: Simplified block diagram of the **RA-HT** spectrum analyzer [P5]

block" is meant an array of *REs*, where the number of *CORDIC REs* is determined by the chosen version of architecture for analyzer-synthesizer (see Table 17). This, in its turn, is determined by the number of *FOR loops*. The use of loops:

- eases the implementation of *Phi*-transform. For example, for version 1 (see Table 17) only one clock pulse is needed to get the spectrum,
- increases the number of used *LEs*, because the transform is spread over the parallel *FPGA* structure.

Summary

Obtained results

- The first experimental *FPGA*-based 1-D **CRAIMOT** spectrum analyzer with serial architecture.
- Three experimental *RE*-based *FPGA* versions of **RA-HT** spectrum analyzers with parallel architecture. In [P5] the *RE* array is labeled as *CORDIC* block.

Table 17: Comparison of analyzer/synthesizer versions [P5]

	FOR loops	*CORDIC cells*	*Logic elements*	*Calculation time (ns)*
1.	2	$n_C = n \cdot 2^{n-1}$	$\approx 800 + 600 \cdot n_C$	$\approx 70 \cdot n$, $(n \leq 3)$
2.	1	$n_C = 2^{n-1}$	$\approx 800 + 600 \cdot n_C$	$\approx 70 \cdot n$
3.	0	$n_C = 1$	$\approx 700 + 600 \cdot n_C + 50 \cdot n \cdot 2^{n-1}$	$\approx 70 \cdot n \cdot 2^{n-1}$

Table 18: The list of parameters of *SANSYN* [P5]

Parameter	*Value*
External clock used	$f_c = 50(100)$ MHz
Maximal number of angles	$n_\varphi = n \cdot N/2 = 5 \cdot 16 = 80$
Maximal block length (samples)	$N = 2^n = 32$
Wordlength for sample values	$w = 10$ bits, Q1.x FPA
CORDIC rotation time	$T_{CORDIC} = 70$ ns
Sample (also processing) time	$T_s = n \cdot T_{CORDIC}$

Conclusions

- Performance of signal spectrum analyzer based on *BF* generator is poor, but can be practically useful, for example, for rapid-analysis of telephony (Fs = 8kHz) or biomedical signals (for example, *EEG*).
- Operation of implemented analyzer with serial architecture is spread in time (see Section 2.2.2). For that reason, a relatively small number of *LEs* is used and the analyzer is useful for low power devices.
- Since *RE*-based analyzer version is spread over the parallel *FPGA* structure, the *RE*-based signal spectrum analyzer is much faster (≈ 65 times for $n = 5$) than the serial spectrum analyzer.
- It is possible to modify and use the *RE*-based spectrum analyzer as **RABOT** (or some **RABOT** subclass, e.g., **CRAIMOT**) spectrum analyzer also.
- Both serial and parallel architectures are useful not only for spectrum analyzers but for synthesizers also.
- It has been experimentally ascertained that implemented analyzer/synthesizer version 1 (using two *FOR loops*) has the highest performance, but it needs the largest number of *REs* and *LEs*. Taking into account that the system performs both spectrum calculation and signal reconstruction, the number of used *LEs* is equal to $\approx$ 16000, if $n = 3$ (for version 1). Version 3 is the most economic in regard to resource consumption (≈ 3800 *LE* for $n = 3$), but it is also the slowest. For version 2 (one *FOR loop*), which may be regarded as tradeoff between the number of *LEs* and implementation complexity, are needed ≈ 6400 *LEs*. The number of *LEs* grows rapidly by increasing n, for example, version 1 is not implementable on the chosen *FPGA* (*EP2C35F672C6*), if $n = 4$.

4.3 Signal analyzer-synthesizer

The virtual speech analyzer-synthesizer (*SANSYN*) (see Figure 4.7) based on rotation angle transforms was developed within the framework of author's master thesis [14]. *SAN-*

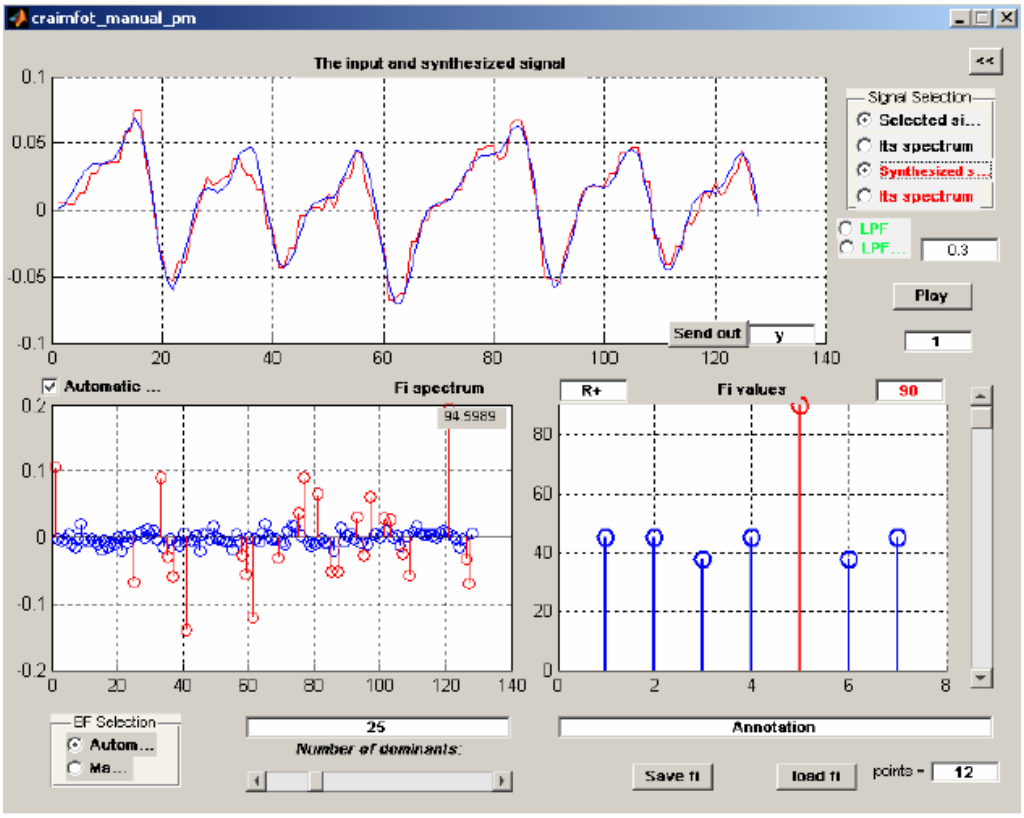


Figure 4.7: Example of speech signal compression

SYN has been firstly mentioned in [50]. This tool is included in the corresponding MatLab Toolbox [P8]. Additionally, *SANSYN* is an essential help for the prototyping of experimental *FPGA* devices, e.g., for the implementation of experimental spectrum analyzer [P2]. In the depicted tool, the search for optimal parameters (angles) is performed half-automatically, by testing the angles one by one. In the latest versions of the developed tool is built-in also nonlinear optimization function that allows to find the optimal transform automatically. In the depicted example the spectrum calculation for Latvian vowel "U" is performed in domain of angle-based transforms. For optimal angles, 95% of signal energy are concentrated in 20% of *BFs* (25 out of 128). The developed tool allows also to show and play an audio signal that is synthesized from spectral coefficients. The tool allows to operate not only with audio but also with any other acoustic signals, including also biomedical signals [51]. Time diagrams and spectrums obtained using the tool are presented in [P1], [P2], [P8] and [3].

Summary

Obtained results

- A virtual tool for acoustic signal analysis and synthesis. It is included in the corresponding *MatLab Toolbox* [P8] (**phiansyn**).

Conclusions

- Parts of the developed tool are used for the building of *MatLab Phi*-transform toolbox [P8]. The tool is very important for the implementation of *FPGA* devices, because it allows to implement into *FPGA* different transforms, get spectrums and synthesize signals.
- A function is built in the latest versions of the tool that allows to find the optimal *Phi*-transform for chosen signal and offers new possibilities in signal compression.
- After finding of optimal angles, it is possible to get a good quality (by subjective assessment) speech signal using a relatively small number of *BFs* ($\approx 20\%$) [P1], [P2], [P8]. It is approved also by experiments with FPGA synthesizer (the material is not yet published).
- The modularity and open structure of the tool makes it a handy help for the prototyping of *FPGA* devices in future.

4.4 Haar-Like filters

In papers [P3], [P6] and [P8] novel orthogonal filters are described. The structure of these filters are alike the structure of orthogonal wavelet decomposition-reconstruction filters. The main difference is that the filters described here are tunable using parameters (rotation angles). In the classical wavelet filters some specific orthogonal wavelet transform is used, but in the novel filters can be used practically an infinite number of changeable transforms. The filters can be run in extraction/rejection, spectrum analyzer/synthesizer and in other modes. Probably, in [P6] and [P8] is firstly described the Signal Shape Resonance (but it is outside the theses for defense). It is possible to extract a chosen signal ideally, if the filter is tuned to selected signal shape, this signal is orthogonal to distortion signal and the signal amplitude is greater than rounding error. In the case of rejection, the amplitude of undesired signal may be maximal but it must be under saturation level. Filtration quality depends on the orthogonality level of signals to be separated [P8]. In the mentioned papers [P3], [P6] and [P8] three different examples are given for illustrative purposes:

- pulse-like signal extraction from additive noise,
- pulse-like signal extraction from the masking pulse train,
- rejection of pulse-like distortion from a corrupted sine wave.

The example in Figure 4.8 shows how a pulse-like signal is extracted from an additive noise using rotation-angle-based orthogonal filters. If the pulse-like signal is orthogonal to the noise (such a case appears in real life with a very low probability) and the parameters of the transform are set so that the signal shape coincides with some of transform *BFs*,

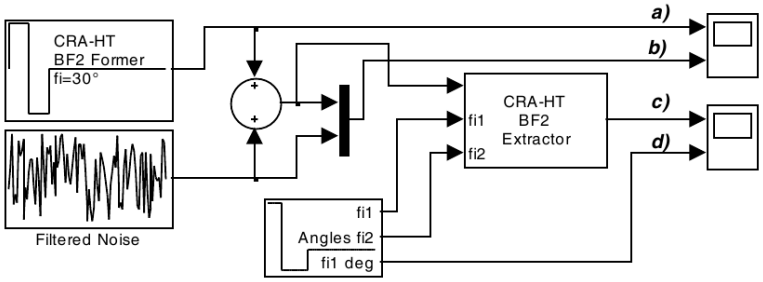


Figure 4.8: Extraction of single 2nd **CRA-HT** *BF* from noise (*SIMULINK* diagram) [P3]

the signal can be extracted from the noise ideally, even if its amplitude is considerably less than the noise amplitude but greater than rounding error. For this demonstration it is artificially ensured that the added pulse-like signal is orthogonal to the noise.

Figure 4.9 shows signal filtering timing diagrams. But Figure 4.10 demonstrates a refinement of sine wave corrupted by a pulse-like signal, when the pulse amplitude exceeds by far the sine amplitude. This example does not provide orthogonality, but the filtering effect is outstanding anyway. The *THD* for the corrupted sine is 100% (a specially chosen distortion amplitude), but after the filtering it falls down to 0.0004%! Whereas, we obtain 12.5% for the Haar wavelets and it is thousand times worse. In the figure are shown simulation data and oscillograms obtained from *FPGA*.

The developed filter can be used as an efficient tool for the real-time pulse extraction/rejection. It is also possible to adapt the filter for non-pulse-like signals, and the filtering quality, as in the case of pulse-like signals, depends only on the mutual orthog-

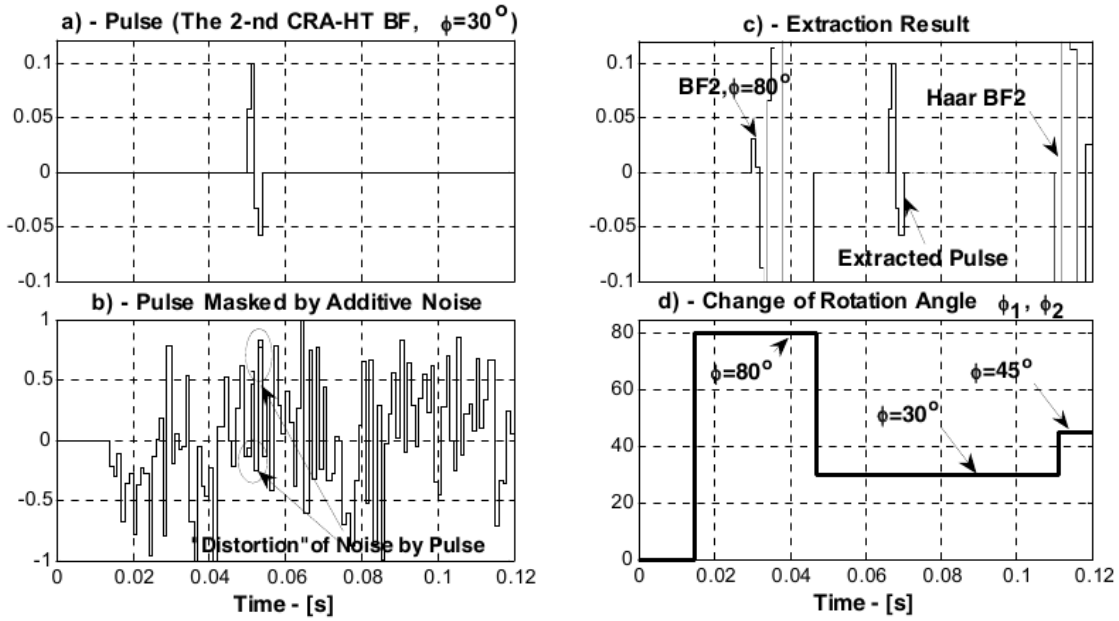
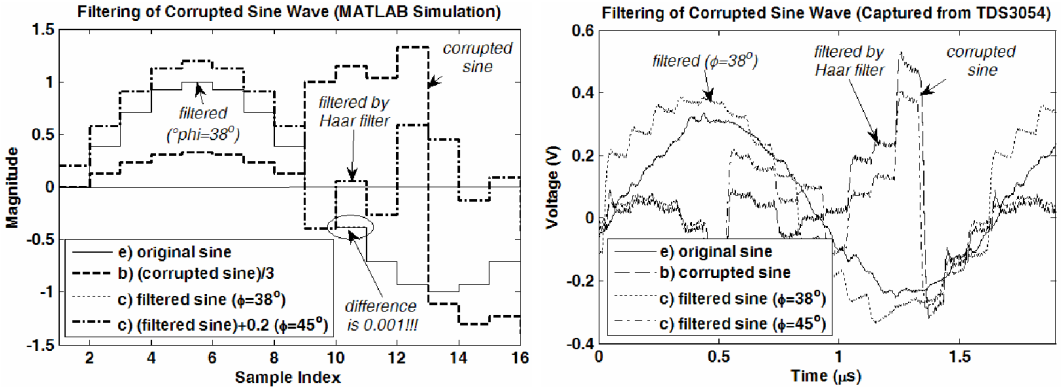


Figure 4.9: Extraction of single **CRA-HT** *BF* from additive noise [P3]



(a) *MatLab* simulation (b) Experimental results

Figure 4.10: Filtering of sine wave corrupted by pulse [P6]

onality level of the signal and distortion. Table 19 summarizes the parameters of *DeRe* filters (for two different implemented versions [P6]) and consumption of *FPGA* resources. The shape of filtered pulses is determined by the filter parameters (angles). By modifying the depicted **CRA-HT** filter, it is possible to get the **CRAIM-HT** filter (see Section 1.2.3), which operates with a larger number of control parameters and, for that reason, the diversity of filtered pulse shapes is larger, too. In our opinion, in the case of **CRAIM-HT** we have the best ratio between the diversity of pulse shapes un the number of parameters.

Table 19: The list of parameters of *DeRe* filter [P6]

Parameter		*Value*
External clock used		$f_c = 50(100)$ MHz
Wordlength for sample values		$w = 10$ bits, *Q1.x FPA*
Sample time (processing time per stage)		$T_s = 70$ ns
Maximal number of stages	version 1	$n = 3$
	version 2	$n = 9$
Maximal number of angles	version 1	$n_\varphi = n \cdot N/2 = 12$
	version 2	$n_\varphi = (2^{n+2} - 2) = 1022$
	v.2, CRA-HT	$n_\varphi = 1$
	v.2, CRAIM-HT	$n_\varphi = n$
	v.2, RSA-HT	$n_\varphi = 2^{n-1}$
Number of logic elements	version 1	$700 \cdot (2^{n+2} - 4)$
	version 2	$1400 \cdot n + 15 \cdot (2^{n+1} - 2)$

Summary on papers [P3], [P6] and [P8]

Obtained results

- *FPGA*-based experimental parametrical orthogonal **RA-HT** filters.
- A set of parameters for *FPGA* implementation of two parametrical filter architectures.

Conclusions

- Literature analysis shows that the developed parametrical filters are original designs.
- The decomposition part of the created filters can be used also as a spectrum analyzer and the reconstruction part – as a signal synthesizer in the domain of corresponding transform.
- The created filters are similar to the classical wavelet filters. The difference between them are such that a particular orthogonal transform for wavelet filters is fixed, but for phi-filters – changeable.
- The experiments with implemented **CRA-HT** and **CRAIM-HT** show that these filters are very effective for extraction/rejection of different pulse-like signals. The filtration efficiency depends on the mutual orthogonality between filtered signal and noise. So, for example, the *THD* for a corrupted sine wave (a pulse-like signal is used for distortion) could be reduced more than 200000 times, and it is 30000 times better than using a wavelet filter.
- Modification of the created filters allows to use them also for filtering signals having in time (space) spread energy (for example, signals with **CRAIMOT** *BF* form). The filtering efficiency depends on the orthogonality level of signals to be separated.
- It is found that the processing time for one sample practically does not depend on the clock frequency, when phi-filters are implemented in *FPGA* using enrolled *CORDIC* algorithm ([P4]-[P7], [P9]). The processing time is $\sim$ 70 (40) ns (for 9 iterations) when implemented on *EP2C35F672C6*.
- Parameters of two *FPGA* implementation architectures (depending on the number of *REs*) are collected. It is found that minimal consumption of *FPGA* resources (for $T_s = 70$ns) is obtained using version 2 architecture [P6].
- Approximate experimental equations for *LEs* consumption assessment are found (see Table 19). For example, if the number of filter stages $n = 3$, then *LEs* consumption for ver. 1 is $\sim$ 19600, but for ver. 2 – $\sim$ 4320.

- It is found that although ver. 1 is easier to make, the consumption of *LEs* is so large, that it is possible to implement 3 times less filter stages than for ver. 2, but the timing performance is comparable.

- The use of several *VHDL FOR cycles* (ver. 1) facilitates the development of *Phi*-filter, because in this case the result is obtained on every clock (the design is spread over parallel *FPGA* structure, when *VHDL FOR* cycles are used). The reduction of the number of *FOR cycles* reduces also the number of used *LEs*, but complicates the management of *Phi*-filter, because the operation of filter is spread in time.

- No multipliers (*DSP* elements) are required, when the *CORDIC* algorithm is used. This allows the *Phi*-filter implementation on lower-cost *FPGAs* (for instance, *Cyclone I*), at the cost of the reduction of timing performance.

4.5 The prototype-simulator of data transmission system based on the generalized orthogonal nonsinusoidal division multiplexing

In data transmission systems orthogonal frequency (sinusoidal) division multiplexing (*OFDM*) is widely used, where the bits[10] of data block (the length of the block depends on a particular *OFDM*) are modulated by a sinusoidal signal that is orthogonal to all other (also sinusoidal) signals. This modulation scheme is preferable for data transmission in distorted channels (mainly because of frequency selective fading). This modulation is performed using *IFFT*. Nonsinusoidal signals also, for instance, different wavelets [47], may be used for modulation. In literature such systems are named as Orthogonal Wavelet Division Multiplexing (*OWDM*) [48], [49]. [53] firstly compares classical *OFDM* and division multiplexing system based on nonsinusoidal orthogonal functions (Hadamard). Japanese researcher Oka [54] also describes the use of nonsinusoidal orthogonal transforms in data transmission systems. Now, this research proves that it is possible to modulate data by nonsinusoidal signals using parametrical transforms based on rotation angles. Since transforms are parametrical, it is possible to change the shape of nonsinusoidal signal, and in such a way also the sinusoidal spectrum structure of transmitting signal before the up-converter.

FPGA implementation of such a system (including implementation of data transmission channel) allows to simulate transmission of very long binary data sequences, which is more time-consuming if other simulation environments (e.g. *Simulink*) are used. The major drawback of this system – not only the angle-based transform should be implemented into *FPGA*, but also the signal generator (*PN Gen* in Figure 4.11), measurement blocks (*BER* counter and Power calculator), digital modulation elements (*QPSK mod* and *QPSK demod*), transform control (ϕ) and data transmission channel (*AWGN channel*). On the other hand, the major advantage of the system implemented in *FPGA* – simulation is

[10] usually, bits are modulated using some of quadrature modulation schemes (*BPSK*, *QPSK*, *4QAM*, etc.)

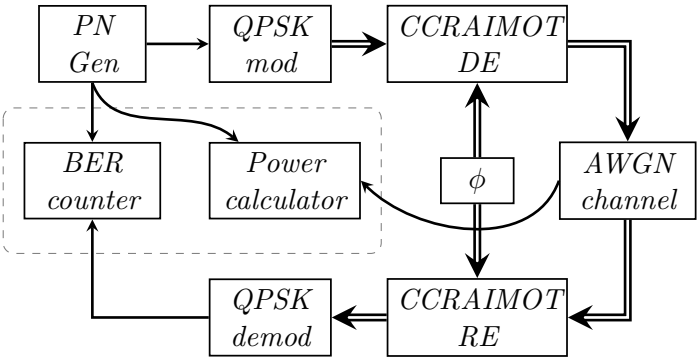


Figure 4.11: Simplified test scheme of digital part of **CCRAIMOT** *GONDM* [P10]

performed in real time at 10 MHz clock frequency (for pn generator), and this means that only one second is needed to get the Bit Error Rate (*BER*) with magnitude 10^{-6}.

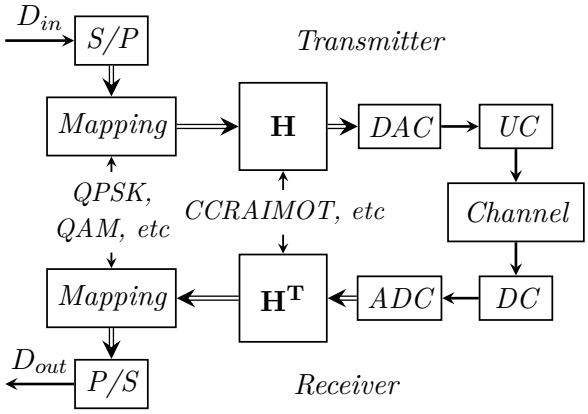


Figure 4.12: Simplified block diagram of *GONDM* data transmission system [P10]

For simulation two kinds of data transmission channels have been developed – a channel with additive white Gaussian noise (*AWGN* channel) and combined channel with flat

Table 20: The list of parameters of design (*Stratix III EP3SL1501152C2*) [P10]

Parameter	*Value*
External clock used	f_c = 80 MHz
Sample (also processing) time	20 ns
Size of data block (N)	64
Wordlength for sample values	w=11 bits, Q1.x FPA
Minimal step of angle	1°
Hardware resources	
DSP blocks (18-bit)	384(100 %)
Logic Utilization	25 %

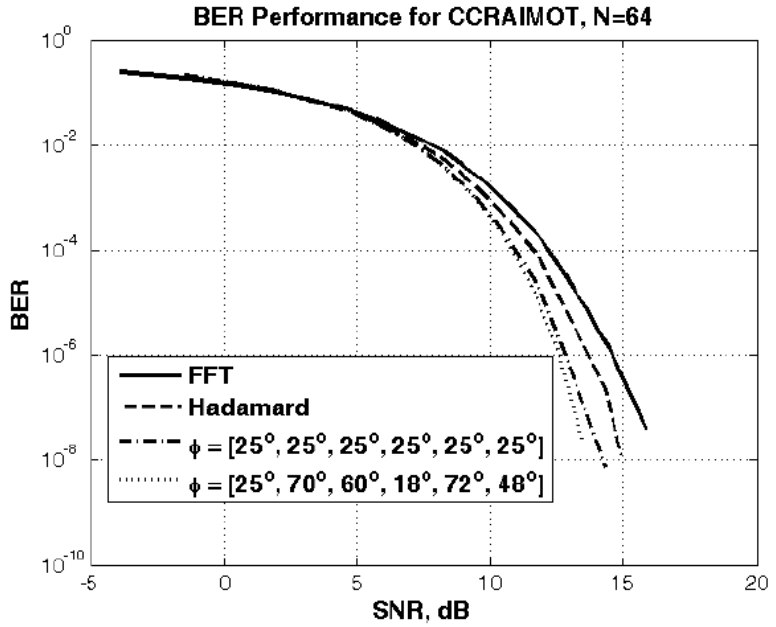


Figure 4.13: Comparison of *BER* performances for different transforms and *AWGN* channel [P10]

fading (*FFC*) and *AWGN*. By combining impacts of several *FFCs* a more complex channel (*FSFC*) can be obtained, but then more sophisticated equalization algorithms would be needed. The whole operation of system and the *BER* calculation are not possible without these algorithms. The classical *OFDM* equalization and synchronization algorithms [52] do not work in the case of *GONDM* [55].

As seen in Figure 4.13, the *BER* performance in the case of *AWGN* channel differs for different transforms.

The Table 20 summarizes the parameters of the developed data transmission system.

Summary on papers [P10]

Obtained results

- Simulator-prototype of *FPGA*-based data transmission system based on generalized orthogonal nonsinusoidal division multiplexing (*GONDM*). It includes:
  - A modulator based on **CCRAIMOT** functions,
  - A transmission channel simulator with additive white Gaussian Noise (*AWGN*),
  - A combined channel simulator, which includes both forms – *AWGN* and flat fading channel (*FFC*),
  - A pseudo noise generator (*pn-generator*),
  - Different auxiliary modulators (*QPSK*, *QAM*, etc.),
  - A power calculator and a *BER* counter.

- Provisional experimental *BER* curves have been obtained using the developed simulator-prototype.

Conclusions

- It is easy to use the DEcomposition/REconstruction filter architecture in *GONDM* modulator, which minimizes *LE* consumption in comparison to parallel architecture.
- *GONDM* is a promising alternative for classical *OFDM*, because it significantly (in fact, unlimitedly) extends the variety of orthogonal modulation schemes used in orthogonal division multiplexing. Provisional experiments show, that **CCRAIMOT** functions can be adjusted to transmitted signal and transmission channel, thereby decreasing the *BER* at least 10 times at signal/noise ratio more than 12 dB.
- Due to a relatively large consumption of *FPGA* resources (all *DSP* elements are used for the simulator-prototype (specifically, *EP3S150L1152C2*), but multiplier creation using *LEs* is highly ineffective – as mentioned in conclusions of Section 3.3), the described 64-channel *GONDM* system could be implemented only in relatively high-class *FPGAs* (*Stratix III* and more powerful).
- When implemented in *FPGA*, the system simulation platform allows to obtain *BER* curves with the order of 10^{-6} in a very short time. That is hundreds of times faster than using Simulink (20-40 min) or dozens of times faster than using *Simulink Real-Time Workshop (RTW)*.
- Valuable experience has been acquired when developing the simulator-prototype. It can be generalized and used also in other fields demanding huge computation resources (for example, in materials science). The *FPGA* simulator-prototype is a promising alternative to parallel computers, because it allows to design real-time devices for different uses.
- For the development of full *GONDM* system more serious research and financial resources are needed. Here we only touch a part of potentially needed researches. Actualities for the nearest future (the work has already started):
  - Synchronization design for data transmission system,
  - Frequency selective fading channel (*FSFC*) simulation,
  - Channel equalization algorithms,
  - Parameter adjusting algorithms.

Conclusion

The subject of phi-transforms is so broad that large human and financial resources are required to fully cover it. The current development in this field has the following specific features:

- Both the phi-transform theory and the possibility of practical implementation are being researched simultaneously.
- Several development directions can be distinguished: signal analysis/synthesis, signal compression, generalized frequency division multiplexing, image processing, automation of development of practical devices, etc.

The current phase of phi-transform development concerns preparing the ground for further research. This doctoral study should be regarded as accomplished and seen as a start platform for more extensive work.

The main emphasis of the doctoral study is laid upon the assessment of the possibility of practical implementation of transforms. The doctoral thesis discusses the implementation in *FPGA* of several different-purpose phi-transform-based devices and provides the comparison of various parameters for them. The described experimental devices are first phi-transform-based devices that have been developed, and they can be regarded as a preliminary platform for developing commercial products. The current experience allows to assess the complexity of phi-transform implementation and improve synthesis algorithms, and to assess and improve device parameters – reducing the number of used *FPGA* logic elements and increasing performance.

An essential preliminary work is done in the field of orthogonal filters, allowing to create parametrical filters with unique properties, and it can be the ground for studies (including doctoral theses) in various fields, for example, in radar and sonar signal processing. For potential commercial purposes, as it mainly concerns creating an alternative to *OFDM* data transmission, the developed experimental prototype-simulator of generalized frequency division multiplexing system is very important. It can serve as a preliminary platform for the development of innovative frequency division multiplexing systems and, at the same time, as an alternative simulation environment to virtual simulators. A preliminary work done in generalized frequency division multiplexing is potentially useful not only for data transmission and digital broadcasting, but also for already mentioned radars and sonars. Research on the issues of this modulation has induced intensive research on the basic element of the generalized transform (*EGURM*).

The notation of *EGURM* allows to describe in a rather simply way all possible structures of Jacobi matrix. As a result, an automated system for *RE* synthesis has been created. By combining the theoretical side (the choice of *EGURM* structures, transformation of mathematical relations, etc.) and the practical side (*VHDL* simulation, estimation

of necessary *FPGA* resources), the way of synthesis of phi-transforms has been fundamentally changed – using the mentioned system, synthesis of basic elements (*REs*) of phi-transforms is done automatically. For the system to be fully automatic, the algorithm for calculation of signal sample wordlengths must be improved and incorporated into the tool for fixed-point arithmetic (*FPA*) error estimation. Currently, the tool allows to calculate the error for different wordlengths of *RE* signal samples, but these values must be entered manually. The improvements are needed to allow to obtain wordlength values for a given error value.

As future work, it is important to develop the automated phi-transform synthesis system *UNITIT*. This system would combine tree-like decomposition-reconstruction algorithms and *EGURIT*. Besides, in the near future, *EGURIT* must be supplemented with the complex *CORDIC* and a resource estimation algorithm. It would allow to use both the built-in traditional rotation algorithm and *CORDIC*, and therefore optimize consumption of logic elements and performance. No less important task is to adjust *EGURIT* and *UNITIT* systems for other development platforms (*Xilinx*, *Synopsis*, etc.), and design corresponding experimental *ASIC* chips. The work on that has already started.

The subject of phi-transforms is open to intensive further research. The development of *FPGA* implementation of phi-transforms for 2-D signals has been started. The first paper on this subject ([11] from the all published papers list) deals with the issues of memory management and implementation architectures for 2-D transforms based on *RE*.

References

[1] "http://digi.lib.ttu.ee"

[2] RTU, "RTU Doktorantura 2008./2009." RTU izdevnieciba, Riga-2009.

[3] M. Tērauds, "Jauna veida diskrētie ortogonālie pārveidojumi un ar signālu apstrādes pielietojumiem saistītās kļūdas", disertācija, RTU, ETF, 234 lpp, 2009.

[4] Gene H. Golub, Charles F. Van Loan, "Matrix computation", The Johns Hopkins University Press, 3rd edition, 1996. - 728 p.

[5] F. Lorenzelli and K. Yao, "SVD updating for nonstationary data," IEEE Catalog Number 0-7803-2123494, 1994, pp.450-459.

[6] J. Gotze, S. Paul, M. Sauer, "An Efficient Jacobi-like Algorithm for Parallel Eigenvalue Computation", IEEE Transactions on Computers, Sept. Vol. 42, Issue 9, pp. 1058-1065, 1993.

[7] H. Toda, Zhong Zhang, "Perfectly Translation-Invariant Complex Wavelet Packet Transforms", International Conference on Wavelet Analysis and Pattern Recognition, 2009. ICWAPR 2009. pp. 374-389, 2009.

[8] Xiao-Ping Zhang, Mita D. Desai and Ying-Ning Peng, "Orthogonal Complex Filter Banks and Wavelets: Some Properties and Design", IEEE transactions on signal processing, VOL. 47, NO. 4, april 1999, pp.1039-1048.

[9] A. M. Trahtman, "Osnovi teorii diskretnih signalov na konecnih intervalah", Moskva:Sovetskoe radio, 1975.

[10] B. J. Fino, R. V. Algazi, "A unified treatment of discrete fast unitary transforms," SIAM J. Comput., 1977, 6, No. 4, pp. 700-717.

[11] P. P. Vaidyanathan, "A unified approach to orthogonal digital filters and wave digital filters, based on LBR two-pair extraction," IEEE Transactions On Circuits And Systems, Vol. CAS-32, No. 7, pp. 673 686, July 1985.

[12] P. Rieder, J. Goetze, J. A. Nossek, and C. S. Burrus, "Parameterization of orthogonal Wavelet transforms and their implementation," IEEE Transactions On Circuits And Systems—II: Analog And Digital Signal Processing, Vol. 45, no. 2, February 1998, pp. 217-226.

[13] H.C. Andrews, J. Kane, "Kronecker Matrices, Computer Implementation, and Generalized Spectra", Journal of the ACM-1970-April-Vol 17, pp. 260-268, 1970.

[14] G. Valters, "CRAIMOT funkciju izmantošana latviešu valodas runas analīzē un sintēzē", maģistra darbs, RTU, ETF, 82 lpp, 2007.

[15] P. Misans, "Introduction Into The Haar Like Transforms Based On Rotation Angles." Scientific Proc. of Riga Technical University, Telecommunications and Electronics, Riga, RTU, vol. 7, Dec., 2007, pp. 6-13.

[16] B.J. Fino, V.R. Algazi, "Unified Matrix Treatment of the Fast Walsh-Hadamard Transform", IEEE Transactions on Computers, pp. 1142-1146, 1976.

[17] J. W. Cooley, J. W. Tukey, "An algorithm for the machine calculation of complex Fourier series", Math. Comput. Vol. 19, No. 90. Apr. 1965, pp. 297–301.

[18] W.H. Chen, C.H.SMith, and S.C.Fralick, "A fast computational algorithm for discrete cosine transform", IEEE Transactions on Communications, Vol.25, pp. 1004-1009, 1977.

[19] C. V. Rammamoorthy, H. F. Li, "Pipeline Architecture", Computing Surveys, Vol. 9, No. 1, March, pp.42, 1977.

[20] R. van Spaendock, T. Blu, R. Baraniuk, M. Vetterli, "Orthogonal Hilbert Transform Filter Banks and Wavelets", Conference on Acoustics, Speech and Signal Processing, Proceedings ICASSP'03, Vol.6, pp. VI-505-8, 2003.

[21] Cishen Zhang, Song Wang and Lihua Xie. "Sequentially Operated FIR Multirate Filter Banks", 5th International Conference on Signal Processing, Proceedings of ISCP200, Vol.1, pp.133-138, 2000.

[22] C. Herley and M. Vetterli, "Orthogonal Time-Varying Filter Banks and Wavelet Packets", IEEE Transactions on signal processing, VOL. 42, NO. 10, October 1994

[23] Guangyu Wang, "The Most General Time-Varying Filter Bank and Time-Varying Lapped Transforms", IEEE Transaction on Signal Processing, Vol.54, No.10, pp.3775-3789, October 2006,.

[24] S. White, "Digital Signal Processing: A Filtering Approach", Delmar Cengage Learning, ISBN-13: 978-0766815315, pp. 256, 2000.

[25] Soo-Chang Pei, Jong-Jy Shyu, "Complex Eigenfilter Design of Arbitrary Complex Coefficient FIR Digital Filters", IEEE transactions on Circuits and Systems, Analog and Digital Signal Processing, Vol. 40, No. 1, pp. 32 - 40, 1993.

[26] U. Meyer-Baese, "Digital Signal Processing with Field Programmable Gate Arrays", Springer, ISBN 3-540-21119-5, pp.527, 2003.

[27] R. C. Ismail, R. Hussin, "High Performance Complex Number Multiplier Using Booth-Wallace Algorithm", IEEE International Conference on Semiconductor Electronics,ICSE '06, Pp. 786 - 790, 2006.

[28] U. Hatnik, S Altmann, "Using ModelSim, Matlab/Simulink and NS for Simulation of Distributed Systems", Proceedings of the International Conference on Parallel Computing in Electrical Engineering (PARELEC'04), pp. 114-119, 2004.

[29] K. Arshak, E Jafer, C Ibala, "Power Testing of an FPGA based System Using Modelsim Code Coverage capability", IEEE Design and Diagnostics of Electronic Circuits and Systems, DDECS '07, pp. 1-4, 2007.

[30] B. Gsetner, David V. Anderson, "Automatic Generation of ModelSim-MatlabInterface for RTL Debugging and Verification", 50th Midwest Symposium on Circuits and Systems, MWSCAS 2007, pp. 1497-1500, 2007.

[31] Sunil R. Das . Jun-Feng Li, Altaf Hossain, Amiya R. Nayak, Emil M. Petriu, Satyendra Biswas, and Wen-Ben Jone, "Improved Test Efficiency in Cores-Based System-on-Chips Using ModelSim Verification Tool", IEEE Instrumentation and Measurement Technology Conference Proceedings, IMTC 2008, pp. 1487-1492, 2008.

[32] Sunil R. Das, Altaf Hossain, Jun -F. Li, Emil M. Petriu, Satyendra N. Biswas, Wen -B. Jone, and Mansour H. Assaf, "Further Studies on Improved Test Efficiency in Cores-Based System-on-Chips Using ModelSim Verification Tool", IEEE Instrumentation and Measurement Technology Conference, I2MTC '09, pp. 1132-1137, 2009.

[33] Yan Li , ling Huo, Xin Li, lin Wen, Yaohui Wang, Bin Shan, "An Open-loop Sin Microstepping Driver Based on FPGA And the Co-simulation of Modelsim and Simulink", 2010 International Conference on Computer, Mechatronics, Control and Electronic Engineering(CMCE), pp. 223-227, 2010.

[34] Marcelo F. Castoldi, Manoel L. Aguiar, "Simulation Strategy in VHDL Induction Motor Control of DTC Code for", 2006 IEEE International Symposium on Industrial Electronics, Vol. 3, pp. 2248-2253, 2006.

[35] Jian Liang; R. Tessier, O. Mencer, "Floating point unit generation and evaluation for FPGAs", 11th Annual IEEE Symposium on Field-Programmable Custom Computing Machines, FCCM 2003, pp. 185-194, 2003.

[36] S. Paschalakis, P. Lee, "Double precision floating-point arithmetic on FPGAs", Proceedings of IEEE International Conference on Field-Programmable Technology (FPT), pp.352-358, 2003.

[37] F. Mayer-Lindenberg, V. Beller, "An FPGA-based floating-point processor array supporting a high-precision dot product", IEEE International Conference on Field Programmable Technology, FPT, pp. 317-320, 2006.

[38] M. Baesler, S. Voigt, T. Teufel, "An IEEE 754-2008 Decimal Parallel and Pipelined FPGA Floating-Point Multiplier", 2010 International Conference on Field Programmable Logic and Applications (FPL), pp. 489 - 495, 2010.

[39] O. Chetelat, "Fixed-Point digital controller", Proceedings of the 2004 American Control Conference, Boston, Massachusetts, pp.2864-2869, 2004.

[40] Bruce W. Bomar, L. Montgomery Smith, and Roy D. Joseph, "Roundoff Noise Analysis of State-Space Digital Filters Implemented on Floating-Point Digital Signal Processors", IEEE Transactions on Circuits and Systems: Analog and Digital Signal Processing, Vol. 44, No. 11, pp. 952-955, 1997.

[41] G. Amit, U. Shaked, "Small Roundoff Noise Realization of Fixed-Point Digital Filters and Controllers", IEEE transactions on Acoustic, Speech and Signal Processing, Vol. 36, No. 6, pp. 880-891, 1988.

[42] "`www.altera.com/products/devices`"

[43] E. L. Oberstar, "Fixed-Point Representation and Fractional Math", Oberstar Consulting, 07/17/2004, 9 pages.

[44] Altera, "Stratix II Performance and Logic Efficiency Analysis", "`www.altera.com/literature/wp/wpstxiiple.pdf`"

[45] M. Otte, M. Bucker, J. Gotze, "Complex Cordic-Like Algorithms for Linearly Constrained MVDR Beamforming", International Zurich Seminar on Broadband Communications, pp. 97-104, 2000.

[46] R. Andraka, "A survey of CORDIC algorithms for FPGA based computers", FPGA'98, Proceedings of the 1998 ACM/SIGDA sixth international symposium on Field programmable gate arrays.

[47] A. R. Lindsey et al., "Wavelet packet modulations: A generalized method for orhogonally multiplexed communications", in Proc. SSST'95, 1995.

[48] W. Yang, G. Bi, T.-S. P. Yum, "A multirate wireless transmission system using wavelet packet modulation", in Proc. 1997 IEEE 47^{th} Vehicular Technology Conference, 1997, vol.1, pp. 368-372.

[49] S.L. Linfoot, M.K. Ibrahims, "Flexible modulation for digital terrestrial broadcasting", Electronic Lett., vol 42, no. 23, Nov. 2006, pp. 1360-1362.

[50] P. Misans, M. Terauds, G. Valters, T. Sile, M. Buikis, "Introduction Into the Fast Orthogonal Transforms Based on Rotation Angles – Part 2: On Phi-function based Signal Analysis," presented at the 10th International Conference Electronics, May 23-25, 2006, Kaunas, Lithuania.

[51] N. Vasilevskis, P. Misans, " Using of Novel Haar Like Transforms for the Detection of Epileptic Spikes," presented at the 12th International Conference Electronics, May 18-20, Kaunas, Lithuania.

[52] H. Sari, G. Karam, I. Jeanclaud, "Frequency-domain equalization of mobile radio and terrestrial broadcast channels", IEEE Global Telecommunications Conference, 1994, GLOBECOM '94, pp. 1-5, 1994.

[53] P. Misans, M. Torkelson, "Preliminary Simulation of Multicarrier Modulation Data Transmission System," Nordic MATLAB Conference '95, Stockholm, Oct. 31 Nov. 1, 1995, Conf. Proc, pp. 55-58.

[54] I. Oka, M. P. C. Fossorier, "A general orthogonal modulation model for software radios," IEEE Transactions on Communications, vol.54, No. 1, Jan, 2006, pp. 7-12.

[55] A. Aboltins, "Comparison of Orthogonal Transforms for OFDM Communication System", Kaunas, 2011, 5, pp. 77-80.