RĪGAS TEHNISKĀ UNIVERSITĀTE

Gatis VALTERS

PARAMETRISKAJIEM ORTOGONĀLAJIEM PĀRVEIDOJUMIEM ATBILSTOŠU SIGNĀLU CIPARAPSTRĀDES EKSPERIMENTĀLU IERĪČU ĪSTENOŠANA PROGRAMMĒJAMAJOS LOĢISKAJOS MASĪVOS

Kopsavilkums par tematiski vienotu publikāciju kopu

Rīga 2011

RĪGAS TEHNISKĀ UNIVERSITĀTE
Elektronikas un Telekomunikāciju fakultāte
Radioelektronikas institūts

Gatis VALTERS
Doktora studiju programmas "Radioelektronika" doktorants

PARAMETRISKAJIEM ORTOGONĀLAJIEM PĀRVEIDOJUMIEM ATBILSTOŠU SIGNĀLU CIPARAPSTRĀDES EKSPERIMENTĀLU IERĪČU ĪSTENOŠANA PROGRAMMĒJAMAJOS LOĢISKAJOS MASĪVOS

Kopsavilkums par tematiski vienotu publikāciju kopu

Zinātniskais vadītājs
Dr. sc. ing., profesors
P. MISĀNS

Rīga 2011

UDK 621.391:517.44(043)
Va 434 p

Valters. G. Parametriskajiem ortogonālajiem pārveidojumiem atbilstošu signālu ciparapstrādes eksperimentālu ierīču īstenošana programmējamajos loģiskajos masīvos. Kopsavilkums par tematiski vienotu publikāciju kopu-R:RTU, 2011, -96. lpp.

Iespiests saskaņā ar Elektronikas pamatu katedras 2011. gada 7. septembra lēmumu, protokols Nr.1

Šis darbs izstrādāts ar Eiropas Sociālā fonda atbalstu projektā "Atbalsts RTU doktora studiju īstenošanai".

ISBN 978-9934-10-261-5

PROMOCIJAS DARBS
IZVIRZĪTS INŽENIERZINĀTŅU DOKTORA GRĀDA IEGŪŠANAI
RĪGAS TEHNISKAJĀ UNIVERSITĀTĒ

Promocijas darbs inženierzinātņu doktora grāda iegūšanai tiek publiski aizstāvēts 2012. gada 12. janvārī, plkst. 15:45, Rīgas Tehniskās Universitātes Elektronikas un Telekomunikāciju fakultātē, Āzenes ielā 12. 210. auditorijā.

OFICIĀLIE RECENZENTI

Profesors, Dr.sc.ing. Guntars Balodis
Elektronikas un Telekomunikāciju fakultāte, Rīgas Tehniskā universitāte

Profesors, Ph.D., Dipl.Eng. Peeter Ellervee
Tallinas Tehniskā universitāte, Datortehnikas departaments

Dr.sc.comp. Jevgeņijs Buls
Elektronikas un datorzinātņu institūts, vadošais pētnieks

APSTPRINĀJUMS

Apstiprinu, ka esmu izstrādājis doto promocijas darbu, kas iesniegts izskatīšanai Rīgas Tehniskajā universitātē inženierzinātņu doktora grāda iegūšanai. Promocijas darbs nav iesniegts nevienā citā universitātē zinātniskā grāda iegūšanai.

Gatis Valters ..... (Paraksts)

Datums: .....

Promocijas darbs ir uzrakstīts kā kopsavilkums par publikāciju kopumu.

Saturs

Attēlu saraksts

Saīsinājumi

Saīsinājums	Nosaukums angļu valodā	Nosaukums latviešu valodā
FIR	Finite Impulse Respone	Galīga impulsu reakcija
IIR	Infinite Impulse Response	Bezgalīga impulsu reakcija
RMS	Root Mean Square	Vidējā kvadrātiskā
MSE	Mean square error	Vidējā kvadrātiskā kļūda
HDL	Hardware description language	Aparatūras apraksta valoda
VHDL	Very high speed integrated circuits Hardware Description Language	Ļoti ātru integrēto shēmu aparatūras apraksta valoda
FPA	Fixed Point Arithmetic	Fiksētā punkta aritmētika
DSP	Digital Signal Processing	Signālu ciparapstrāde
LUT	Look Up Table	Loģisko kombināciju shēma
ALUT	Adaptive Look Up Table	Adaptīvā loģisko kombināciju shēma
CORDIC	COordinate Rotation DIgital Computer	Koordināšu rotācijas digitālais algoritms
RTL	Register Transfer Level	Reģistru pārvades līmenis
TPD	Time Propagation Delay	Izplatīšanās laiks
SMT	Symbolic Math Toolbox	Simboliskās matemātikas rīki
ASCII	American Standard Code for Information Interchange	Amerikas kodu standarts informācijas apmaiņai
BF	Basis Function	Bāzes funkcija
ADC	Analog-Digital Converter	Analogais-ciparu pārveidotājs
DAC	Digital-Analog Converter	Ciparu-analogais pārveidotājs
UC	Up Converter	Pārveidotājs uz augšu, modulators
DC	Down Converter	Pārveidotājs uz leju, modulators
QAM	Quadrature Amplitude Modulation	Kvadratūrā amplitūdas modulācija
QPSK	Quadrature Phase-Shift Keying	Kvadratūrā fāzes manipulācijas modulācija
IEEE	Institute of Electrical and Electronics Engineers	Elektro- un elektronikas inženieru institūts
OFDM	Orthogonal Frequency-Division Multiplexing	Ortogonālu signālu frekvenčdale
EGURM	Elementary Generalized Unitary Rotation Matrix	Vispārinātā elementārā rotācijas matrica

LE	Logic Element	*Altera*'s loģiskais elements
FPGA	Field Programmable Gate Array	Programmējamo loģiku masīvs
FT	Fast Transform	Ātrā transformācija
FFT	Fast Fourier Transform	Ātrā Furjē transformācija
IFFT	Inverse Fast Fourier Transform	Inversā ātrā Furjē transformācija
DCT	Discrete Cosine Transform	Diskrētā kosinusa transformācija
AWGN	Additive White Gaussian Noise	Aditīvs baltais Gausa troksnis
FFC	Flat Fading Channel	Kanāls ar visu frekvenču spektra komponenšu vienmērīgu vājinājumu
FSFC	Frequency Selective Fading Channel	Kanāls ar selektīvu frekvenču spektra komponenšu vājinājumu
GONDM	Generalized Orthogonal Nonsinusoidal Division Multiplexing	Vispārinātā ortogonālu nesinusoidālu signālu frekvenčdale
M	Multipliers	Reizinātāji
A	Adders	Summatori
MC	Multiplications by constant	Reizināšanas ar konstanti
SOPC	System On a Programmable Chip	Sistēma uz programmējamas mikroshēmas
EGURIT	Elementary Generalized Unitary Rotation Implementation Tool	Elementārās vispārinātās unitārās rotācijas īstenošanas līdzeklis
UNITIT	UNItary Transform Implementation Tool	Vispārinātā unitārā pārveidojuma īstenošanas līdzeklis
SANSYN	Spectrum ANalyzer-SYNthesizer	Spektra analizators-sintezators

Termini

Termins angļu valodā	Termins latviešu valodā
Rotation element (*rotator*)	Rotācijas elements (*RE*), tekstā minēts arī kā žargons "pagriezējs"
Pipeline	Konveijera
Wordlength	Bitu skaits *FPA* attēlojumā (turpmāk *WL*)
Decimation, Downsampling	Decimācija
Altera, Xilinx	*FPGA* ražotāji
NIOS	*Altera*'s virtuālais procesors

Apzīmējumi

Apzīmējums	Paskaidrojums
Φ_k	Parametru (leņķu) kopums (ϕ_k, ψ_k, γ_k)
$\mathbf{\Phi}$	Parametru (leņķu) matrica
$\mathbf{\Phi}_p$	Parametru (leņķu) matricas p−tā kolonna
$\mathbf{\Phi}_H$	Hārveidīgā pārveidojuma parametru (leņķu) matrica
$\mathbf{T}_U$	Vispārinātā Jakobi rotācijas matrica
$\mathbf{B}_b$	Retināta faktorizētā blokveida rotācijas matrica
$\mathbf{B}_s$	Retināta faktorizētā kāpņveida rotācijas matrica (publikācijās parādās kā $\mathbf{B}$)
$\mathbf{P}$	Permutāciju matrica
$\mathbf{0}$	Nuļļu matrica
$\mathbf{I}$	Vienības matrica
$\mathbf{H}$	Ortogonālā pārveidojuma matrica
Φ	Ortogonālā pārveidojuma matrica (apzīmējums vecākās publikācijās)
$\mathbf{H}_H$	Hārveidīgo pārveidojumu matrica
$(...)^{-1}$	Ermita matrica
$\mathbf{X}, \mathbf{Y}$	Ieejas un izejas signālu vektori
$\delta_{\mathbf{p},\mathbf{k}}$	Kronekera simbols
c	$\cos(\phi)$
s	$\sin(\phi)$
N	Transformācijas matricas izmērs ($N = 2^n$)
n	Pārveidojuma faktorizēto matricu skaits ($n = \log_2(N)$)
$\overline{(...)}$	Kompleksi saistītais lielums
$H(z), H(\omega)$	Filtra pārvades funkcija
ω	leņķiskā frekvence
ϵ_{RMS}	Vidējā kvadrātiskā kļūda
$\widehat{\mathbf{v}}$	Signāla vektors ar fiksētā punkta kļūdu
Fs	Diskretizācijas frekvence
Ts	Diskretizācijas laiks
w	*FPA* bitu skaits
Lo	Aproksimācijas filtrs
Hi	Detalizācijas filtrs

Promocijas darba noformējums

Promociju darbs ir veidots kā kopsavilkums par autora zinātniskajām publikācijām, kas ir tapušas laika periodā no 2007.g. līdz 2011.g. Tā kā autora rīcībā ir relatīvi plašs spektrs ar šāda tipa (pārskats par vienotai tematikai veltītām publikācijām, piemēram, [1]) promocijas darbu paraugiem, tad no promocijas darba noformēšanas viedokļa nebija viegli izvēlēties darba noformēšanas formu, galvenokārt, apjoma dēļ – pārskata daļa bez pašām publikācijām mēdz saturēt no 15 lpp. līdz pat 200 lpp. un, otrkārt tādēļ, ka reglamentējošos dokumentos (apkopoti, piemēram, [2]) nefigurē precīzi definēti noformēšanas noteikumi. Autoraprāt, darbam tika izvēlēts saprātīgs kompromisa noformējums (hibrīds) tā, ka darba:

- sākuma daļa ir aizgūta no klasiskā kopsavilkuma (autoreferāta) ievaddaļas,
- kopsavilkuma aprakstošā daļa ir veidota kā reducēta apjoma klasiskā disertācija vai kā paplašināta klasiska kopsavilkuma (autoreferāta) pamatsatura daļa,
- pielikumā ir pievienotas visas ar aizstāvamo darbu saistītās publikāciju kopijas.

Aprakstošā daļa ir veidota tā, lai, autoraprāt, lasītājs bez iedziļināšanās publikācijās varētu izprast materiālu. Lielākā daļa attēlu, tabulu un formulu ir no publikācijām, taču ir arī papildinājumi ("apraksta nogludināšanai"), kas ir veidojušies laika periodā no publikāciju iesniegšanas brīža līdz kopsavilkuma rakstīšanas brīdim vai palikuši ārpus publikācijām to ierobežotā apjoma dēļ.

Tēmas aktualitāte

Šis darbs ir veltīts **CRABOT**[1] (komplekso, uz rotāciju leņķiem balstīto, - turpmāk lietosim žargonu "leņķisko") pārveidojumu realizācijas problēmām *FPGA*. Lai izvairītos no smagnējiem apzīmējumiem, publikācijā [P8] pārveidojumi ir nosaukti par fī-pārveidojumiem vai fī-transformācijām. Kā jau minēts Māra Tērauda disertācijā [3], "leņķi kā parametri nav jaunums, tie tiek pielietoti Givensa rotācijā, SVD[2] [4], [5] un EVD[3] [6] piemeklēšanas algoritmos." Minētajā disertācijā ir aplūkoti reālie leņķiskie pārveidojumi (**RABOT** un to dažādas apakšklases). Pēdējā laikā, kā to liecina, piemēram, komplekso veivletu pētniecības attīstība [7], ir aktualizējusies arī komplekso leņķisko pārveidojumu izpēte. Kompleksās rotācijas (Jakobi rotācijas) pēdējās desmitgadēs plaši tiek izmantotas SVD un QR-algoritmos (īpašvērtību aprēķinos), bet pēdējā desmitgadē sakarā ar straujo $ASIC$ un $FPGA$ shēmu sarežģītības un ātrdarbības pieaugumu, ir pavērusies reāla iespēja šo algoritmu plašākai iemiesošanai mikroshēmās.

[1] *Complex Rotation Angle Based Orthogonal Transform* - Kompleksie ortogonālie pārveidojumi, balstīti uz rotācijas leņķiem

[2] *Singular Value Decomposition* - Singulāro vērtību dekompozīcija

[3] *Eigenvalue Decomposition* - Īpašvērtību dekompozīcija

Pēdējo sešu gadu laikā RTU ETF profesora P. Misāna vadībā notiek gan teorētiska gan praktiska rakstura pētījumi, kas ir saistīti ar leņķisko pārveidojumu izmantošanas perspektīvām un iespējām dažādu signālu analīzē/sintēzē, kompresijā, filtrācijā, datu pārraidē un attēlu apstrādē [P8]. Lai gan divas no **RABOT** pārveidojumu klasēm (**CRAOT**[1] un **CRAIMOT**[2]) pirmoreiz ir aprakstījis Endrjū (Andrews) kā vispārināto Kronekera matricu apakšklasi (ar nosaukumiem Generalized Kronecker matrices [13]), RTU grupas būtisks pienesums ir jaunu parametrisko pārveidojumu definīcijas metodikas izstrādē un dažādu, gan virtuālo, gan uz *FPGA* balstītu, eksperimentālo ierīču izveidē. Pētāmie pārveidojumi ir parametriski pārveidojumi, kuru aprakstam izmanto vairākus parametrus (rotācijas leņķus). Pārveidojumu parametriskā daba ļauj iegūt ortogonālās bāzes funkcijas ar praktiski bezgalīgu formu dažādību. Tas, savukārt, būtiski paplašina ciparu signālapstrādes iespējas gan analīzē/sintēzē, gan filtrācijā, gan kompresijā, gan datu pārraidē un ļauj konkurēt ar jau labi zināmajiem veivletu pārveidojumiem [P8].

Ja atskatāmies vēsturē, tad par unitāro pārveidojumu vispārinājuma (unifikācijas) pamatlicēju pēc Endrjū zināmā mērā var uzskatīt arī Trahtmanu [9], kurš būtiski paplašina diskrēto signālu analīzē/sintēzē izmantojamo pārveidojumu loku, taču viņš neakcentē rotācijas leņķus kā parametrus. Visbūtiskākais pienesums pārveidojumu vispārinājumos "pirmsveivletu ērā" ir no Fino un Algazi puses [10], kuri formulē vairākus likumus unitāro pārveidojumu sintēzei, taču tāpat izpaliek rotācijas leņķu akcents. Par ortogonālās filtrācijas, kurā ir akcentēta rotācija, pionieri var uzskatīt Vaidanathanu [11], taču viņš, savukārt, neaplūko tos kontekstā ar unitārajiem pārveidojumiem. No veivletu pētniekiem noteikti ir jāatzīmē Rīders ar kolēģiem [12], kurš runā par veivletu parametrizāciju ar rotācijas leņķiem, izmantojot *CORDIC* algoritmu veivletu filtru īstenošanā. Nav šaubu, ka jaunu pārveidojumu izmantošana signālu analīzē/sintēzē un filtrācijā paver jaunas iespējas un to pētniecība ir ļoti aktuāla. Tas pats ir sakāms arī par vispārinātās frekvenčdales izmantošanu datu pārraidē, par kuru pirmoreiz, iespējams, tiek runāts [52].

Izstrādājamo pārveidojumu sarežģītība un realizācija ir praktiski tāda pati kā ātrajam Furjē pārveidojumam [9] vai veivletiem, bet potenciālās izmantošanas iespējas ir nesalīdzināmi plašākas. No vienas puses, lai izpētītu šīs iespējas, aktuāla ir pētījumu virzienu dažādošana (signālu analīze/sintēze, filtrācija, kompresija, datu pārraide, attēlu apstrāde utt.). Savukārt, no pārveidojumu praktiskās īstenošanas puses, lai novērtētu fī-pārveidojuma aparatūriskās realizācijas iespējamību kopumā, ne mazāk aktuāla ir arī eksperimentālo ierīču īstenošanas dažādošana. Bez tam būtiska ir leņķisko pārveidojumu "pamatķieģelīša" – elementārās vispārinātās unitārās rotācijas matricas un tās aparatūriskās realizācijas iespēju pastiprināta izpēte. Pašreizējais mikroelektronikas attīstības līmenis ļauj daudz vienkāršāk aparatūriski īstenot sarežģītus algoritmus nekā tas bija iepriekšējās desmitgadēs. Tas būtiski atvieglo un aktualizē arī uz rotācijas leņķiem balstītu signālapstrādes algoritmu

[1] *Constant Rotation Angle Orthogonal Transform* - Konstanta rotācijas leņķa ortogonāls pārveidojums

[2] *Constant Rotation Angle Inside Matrix Orthogonal Transform* - Ortogonāls pārveidojums ar konstantu rotācijas leņķi vienas faktorizētās matricas ietvaros

iemiesošanu *FPGA*. Tā kā jaunākie *FPGA* čipi satur jau vairākus miljardus transzistoru, tad ļoti aktuāla kļūst signālapstrādes algoritmu realizācijas automatizācija. *FPGA* mikroshēmas atšķirībā no *ASIC* mikroshēmām ir pievilcīgas ar to, ka tās ļauj maksimāli minimizēt laiku līdz ierīču praktiskajai izveidei un testēšanai. Tā kā *FPGA* saimes ir ar dažādu arhitektūru, ātrdarbību, patērējamo jaudu un loģisko elementu skaitu, tad ļoti aktuāls ir minētajiem signālapstrādes algoritmiem nepieciešamā resursu daudzuma (reizinātāju un loģisko elementu skaits) ātrdarbības un jaudas patēriņa novērtējums, atkarībā no nepieciešamās signālapstrādes precizitātes.

Darba mērķis

Šis darbs ir veltīts **RABOT** un **CRABOT** leņķisko pārveidojumu (fī-pārveidojumu) algoritmu īstenošanai *FPGA*. Darba mērķis ir **aplūkot un risināt problēmas, kas ir saistītas ar RABOT un CRABOT pārveidojumu algoritmu praktiskas īstenošanas iespējamību un veikt minēto algoritmu aprobāciju *FPGA***. Lai sasniegtu šo mērķi, tika izvirzīti šādi pamatuzdevumi:

1. Aprakstīt elementāro vispārināto unitāro rotācijas matricu un tās modifikācijas.
2. Izpētīt un aprobēt automatizācijas iespējas elementārās vispārinātās unitārās rotācijas matricas īstenošanai *FPGA*.
3. Izpētīt un aprobēt dažādu, uz rotāciju leņķiem balstītu, bāzes funkciju ģeneratoru arhitektūru piemērotību īstenošanai *FPGA*.
4. Izpētīt un aprobēt dažādu, uz rotāciju leņķiem balstītu, spektra analizatoru arhitektūru piemērotību īstenošanai *FPGA*.
5. Izpētīt un aprobēt, uz rotāciju leņķiem balstītu, parametrisko ortogonālo filtru īstenošanas iespējas *FPGA*.

Pētījuma metodika

Pētījumu metodika ietvēra sevī četrus tipveida soļus: analītiskos aprēķinus, skaitliskos aprēķinus, simulāciju un praktiskos eksperimentus. Algoritmu sagatavošana tālākajai īstenošanai vispirms notika analītiski, manuāli izmantojot matricu algebru un labi zināmas matricu īpašības, kā arī pielietojot *MatLab* simboliskās (analītisko aprēķinu un izvedumu) matemātikas iespējas. Algoritmu prototipēšana vispirms notika, veicot programmēšanu *MatLab* un *VHDL* valodās, modeļu veidošanu *Simulink* vidē, skaitliskos aprēķinus un datorsimulāciju ar *MatLab/Simulink* un *Quartus II/ModelSim* programmatūrām. Eksperimentālā algoritmu īstenošana un eksperimentālo ierīču prototipu testēšana notika, izmantojot Alteras *Quartus II* programmatūru un izstrādes rīkus (Development Kits).

Zinātniskā novitāte un galvenie rezultāti

Promocijas darbā pirmo reizi ir apskatīti jautājumi, kas ir veltīti, uz rotācijas leņķiem balstītu, ātro unitāro pārveidojumu (**CRABOT** transformāciju) realizācijai *FPGA*. Darba galvenās novitātes ir saistītas ar jauno pārveidojumu algoritmu īstenošanu eksperimentālās, uz *FPGA* balstītās, ierīcēs un programmās:

- Pirmo reizi izveidoti eksperimentālie parametriskie *BF* ģeneratori, parametriskie spektra analizatori, parametriskie ortogonālie filtri, kas balstās uz izstrādātajiem darbības algoritmiem un arhitektūrām.
- Pirmo reizi uz *FPGA* izveidots vispārinātās ortogonālās parametriskās frekvenčdales prototips-simulators, kas balstās uz izstrādātā darbības algoritma un arhitektūras.
- Pirmo reizi izveidota automatizētā sistēma elementārā vispārinātā unitārā rotācijas elementa sintēzei *FPGA*, kas balstās uz izstrādātā sistēmas darbības algoritma.

Darbā ir iegūti šādi galvenie rezultāti:

- Nodefinēta elementārā vispārinātā unitārā rotācijas matrica,
- Izveidota *MatLab/Simulink/QuartusII/ModelSim* balstīta automatizētā sistēma elementārā vispārinātā unitārā rotācijas elementa sintēzei *FPGA*,
- Izveidoti eksperimentālie parametrisko ortogonālo filtru *FPGA* paraugi, kas, ar atbilstošo *FPA* precizitāti, veic ortogonālu signālu izdalīšanu/nospiešanu,
- Uz *FPGA* tika izveidots vispārinātās ortogonālās nesinusoidālās parametriskās frekvenčdales prototips-simulators, kurā ir izmantots parametriskais **CCRAIMOT** pārveidojums,
- Izveidoti eksperimentālie parametriskie **CRAIMOT** *BF* ģeneratori,
- Izveidoti eksperimentāli **CRAIMOT** un **RA-HT** parametriskie spektra analizatori.

Aizstāvamās tēzes

Aizstāvēšanai tiek izvirzītas sekojošas tēzes:

1. ka izveidotā oriģinālā automatizētā sistēma elementārā vispārinātā unitārā rotācijas elementa (rotatora) sintēzei, izmantojot izstrādātos algoritmus, būtiski vienkāršo rotatora, uz *FPGA* orientēto, sintēzes procesu, samazinot sintēzes laiku, samazinot loģisko elementu patēriņu un palielinot ātrdarbību;
2. ka izveidotie, uz *FPGA* balstītie, parametriskie *BF* ģeneratori, parametriskie spektra analizatori un parametriskie ortogonālie filtri ar virknes, paralēlajām un kokveida arhitektūrām ļauj izvēlēties optimālus risinājumus, vadoties no ierīču ātrdarbības un loģisko elementu patēriņa;

3. ka izveidotie oriģinālie, uz *FPGA* balstītie, parametriskie ortogonālie filtri ļauj izdalīt signālus ar brīvi izvēlētu formu labāk nekā, piemēram, veivletu filtri;

4. ka izveidotais, uz oriģināla algoritma un *FPGA* balstītais, vispārinātās ortogonālās nesinusoidālās parametriskās frekvenčdales prototips-simulators ļauj būtiski samazināt modelējamās sistēmas simulācijas laiku un prototipēt jauna veida frekvenčdales sistēmas, kas ir daudzsološa alternatīva klasiskajai frekvenčdalei (*OFDM*).

Darba praktiskais pielietojums

Darbam ir praktisks pielietojums, realizējot eksperimentālas „leņķiskās" ierīces *FPGA*. Pašreiz izveidotās eksperimentālās *FPGA* ierīces var kalpot kā prototipi tuvākajā nākotnē veidojamajām ierīcēm, kuru darbības pamatā būs „leņķiskie" algoritmi. Darba praktiskās iestrādes aptver plašu eksperimentālo ierīču loku, sākot ar bāzes funkciju ģeneratoriem, unikāliem ortogonālajiem filtriem un beidzot ar datu pārraides sistēmas prototipu. Īpaša praktiska vērtība ir izveidotajai automatizētajai sistēmai *EGURIT*, kas ļauj sintezēt *FPGA* vairākus tūkstošus elementārā vispārinātā unitārā rotācijas elementa vienkāršojumus (modifikācijas), izmantojot ar formulu uzdotu rotācijas matricu. *EGURIT* sistēma ir būtiska pašreiz izstrādājamās unitāro pārveidojumu īstenošanas sistēmas *UNITIT* sastāvdaļa.

Darba aprobācija

Darba publiskā apspriešana zinātniskajās konferencēs:

- "The 10th International Conference ELECTRONICS", 2006.gadā, 23.-25. maijā, Kauņā, Lietuvā.
- "RTU 48th International Scientific Conference", 2007.gadā, 11.-13. oktobrī, Rīgā, Latvijā.
- "MIXDES-2007", 2007. gadā, 20.-22. jūnijā., Čehočinekā, Polijā.
- "ECS'07, the 6th Electronic Circuits and Systems Conference", 2007.g. 6.-7. septembrī, 2007, Bratislavā, Slovākijā.
- "Norchip-2007", 2007. gadā, 19.-20. novembrī., Elborgā (Alborg), Dānijā.
- "Norchip 2008", 2008. gadā, 17.-18. novembrī, Tallinā, Igaunijā.
- "Nordic MATLAB User Conference 2008", 20.-21. novembrī, Stokholmā, Zviedrijā.
- "Norchip 2009", 2009. gadā, 16.-17. novembrī, Trondheimā, Norvēģijā.
- "The 15th International Conference ELECTRONICS", 2011.gadā, 23.-25. maijā, Kauņā, Lietuvā.

- *“RTU 52th International Scientific Conference”, 2011. gadā, 13.-14. oktobrī, Rīgā, Latvijā.
- *“Norchip 2011”, 2011. gadā, 14.-15. novembrī, Lundā, Zviedrijā.

* Pieņemts publicēšanai

Promocijas darba galvenās tēzes tika publicētas

[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.

Visas ar promociju darbu saistītās publikācijas

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 International 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 publikācijas.

Darba apjoms

Kopsavilkums par publikācijām sastāv no četrām nodaļām, bibliogrāfijas saraksta un noslēguma daļas. Kopsavilkums uzrakstīts latviešu un angļu valodās uz 96 lappusēm, tas kopumā satur 55 attēlus, 20 tabulas un 55 atsauces uz literatūru.

Klasiskā kopsavilkuma (autoreferāta) ievaddaļa aizņem 19. lpp, kopsavilkuma aprakstošā daļa – 77 lpp, bet publikāciju daļa aizņem 70 lpp.

Ievads

Ievērojamu vietu signālu apstrādes algoritmu klāstā ieņem ātrie ortogonālie pārveidojumi. Pēdējā desmitgadē īpaši strauja attīstība notikusi zibšņa jeb veivletu funkciju (*Wavelets*, kas ir viena no ortogonālo funkciju apakšklasēm) jomā. Pēdejo gadu laikā RTU ETF tiek pētīti reāli 1-D diskrētie ortogonālie pārveidojumi. Ir aizsācies darbs pie kompleksajiem 1-D ortogonāliem pārveidojumiem (uzsvars šajā kopsavilkumā) un reālajiem 2-D diskrētiem ortogonāliem pārveidojumiem.

Tā kā promocijas darbs ir veidots kā publikāciju kopums, tad zemāk ir sniegts ļoti īss apraksts par katru no publikācijām. Detalizēta informācija par ortogonāliem pārveidojumiem atrodama kopsavilkuma tekstā, kur 1. nodaļā sniegti uz rotācijas leņķiem bāzētu komplekso ortogonālo pārveidojumu teorētiskās sakarības, kā arī parādīta to dažādības daudzveidība. 2. nodaļā apskatītas dažādas ortogonālo pārveidojumu realizācijas arhitektūras. Ar pārveidojumu realizāciju *FPGA* mikroshēmās saistītās problēmas un to risinājumi apskatīti 3. nodaļā. Disertācijas kopsavilkuma pēdējā, 4. nodaļa, veltīta uz rotācijas leņķiem bāzēto eksperimentālo ierīču apskatam.

Uz leņķiem bāzētie ortogonālie pārveidojumi ļauj iegūt bezgalīgi daudz (bet ne visas iespējamās) dažādas bāzes funkcijas (*BF*), tāpēc, jau maģistra darba [14] izstrādes laikā, tika izveidoti dažādi reālu *BF* ģeneratori [P1]. Paralēli tika izstrādāts arī reāla laika audio signālu vispārinātā spektra analizators [P2]. Hārveidīgā signālu dekompozīcijas–rekonstrukcijas sistēma, kas veidota izmantojot parametriskus ortogonālus filtrus aprakstīta [P3]. Padziļinātu *BF* ģeneratora problēmu risinājumi, kas saistīti ar realizāciju *FPGA* mikroshēmās, publicēti [P4]. RTU zinātnisko rakstu krājumā publicētie raksti [P5] un [P6] apkopo parametrisko, reālo Hārveidīgo pārveidojumu realizāciju *FPGA*. [P6] veikta signālu filtrācija, izmantojot dekompozīcijas–rekonstrukcijas sistēmu. Sistēmu realizācija reālās iekārtās, izmantojot fiksētā punkta aritmētiku (*FPA*), vienmēr saistās ar *FPA* kļūdām. [P7] publicēti dati par *FPA* kļūdām, kas rodas realizējot dažādus *BF* ģeneratora algoritmus. Darbojoties ar jauna veida ortogonālajiem pārveidojumiem, ērtību labad tika izveidota un aprakstīta specializētu funkciju *MatLab* bibliotēka [P8]. Viena kompleksās rotācijas matricas (skat. 1. nodaļu) vienkāršotā gadījuma (skat. 3.4. nodaļu) realizācija *FPGA* apskatīta [P9]. [P10] publicēts nesinusoidālas frekvenčdales pārraides sistēmas ciparu daļas realizācijas apraksts, kam par pamatu ņemts kompleksās rotācijas matricas vienkāršots gadījums. Komplekso rotācijas matricu dažādu struktūru un vienkāršojumu (skat. 3.4. nodaļu) *FPGA* realizāciju automatizācijas rīku izveides nepieciešamība, pamatota [P11]. Savukārt, [P12] aprakstīti automatizācijas rīki un to darbības principi.

1. Kompleksie parametriskie ortogonālie pārveidojumi

Šajā nodaļā apkopotas un vispārinātas teorētiskās fī-pārveidojumu sakarības, kas kopīgas visiem publicētajiem rakstiem [P1]-[P12]. Vispārinātās Jakobi matricas pieraksta ieviešana (skat. 1.2.1. nodaļu) [P11],[P12], ļauj attēlot visus iespējamos fī-pārveidojumu variantus - kā reālus [P1]-[P7], tā kompleksus [P8]-[P12]

1.1. Ortogonālie, ortonormālie un unitārie pārveidojumi

Diskrētos ortogonālos pārveidojumus izmanto, lai aprēķinātu diskrēta signāla spektru.

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

kur H - Ortogonālā pārveidojuma matrica, $(\dots)^{-1}$ - inversā matrica (kompleksajiem pārveidojumiem - Ermita (*Hermitian*) matrica), konstanšu k_1 un k_2 vērtības atkarīgas no konkrētā pārveidojuma.

Ortonormālie pārveidojumi un tiem atbilstošās matricas ir ortogonālo pārveidojumu speciāls gadījums, kurā bāzes funkcijas (matricas rindas) norma ir vienāda ar 1. Tikai šādā gadījumā (1.1) izteiksmēs konstantes $k_1 = k_2 = 1$. Kā labi zināms [4], kompleksās ortonormālās matricas tiek dēvētas par unitārajām matricām, bet tām atbilstošos pārveidojumus par unitārajiem pārveidojumiem. Veicot matricas rindu skalāro reizinājumu, varam pārliecināties, vai izvēlētā matrica ir ortonormāla, jo ortonormālām matricām spēkā ir sekojoša sakarība:

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

kur $\delta_{p,k}$ – Kronekera simbols, H_m – H matricas m-tā rinda,
$H_{m,q}$ – H matricas m-tās rindas q-tais elements.

1.2. Uz rotācijas leņķiem bāzēti unitārie pārveidojumi

Šajā nodaļā tiks aplūkoti uz leņķiem bāzēto komplekso ortonormēto (unitāro) pārveidojumu pamatelementi un sniegts neliels priekšstats par to iedalījumu. Sīkāk par reālu uz leņķiem bāzētu ortogonālo pārveidojumu iedalījumu var atrast [3] un [14].

1.2.1. Elementārā vispārinātā unitārā rotācijas matrica

Jaunā veida parametrisko unitāro pārveidojumu pamatelements ir elementārā vispārinātā unitārā rotācijas matrica (turpmāk - elementārā rotācijas matrica), kas ir labi zināmās [4] Jakobi rotācijas matricas vispārinājums. Kompleksu signālu gadījumā, elementārā rotācijas matrica ir 3 parametru (ϕ, ψ un γ) matrica. Ja parametri ϕ,ψ,$\gamma \in [0,45^\circ]$, tad iespējamas 64 dažādas rotācijas matricas struktūras. Rotācijas matricu struktūru skaits

samazinās divas reizes (līdz 32-ām), ja ϕ,ψ,$\gamma \in [0{,}90°]$. Dažas no iespējamajām rotācijas matricas struktūrām parādītas 1. tabulā. Lai salīdzinoši ērti būtu iespējams aprakstīt visas rotācijas matricas struktūras, tiek ieviests vispārinātās Jakobi matricas apzīmējums [P12]:

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

Ieviešot apzīmējumus (1.4), matricu (1.3) varam pārrakstīt kā (1.5) [P11].

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

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

Tā, piemēram, ņemot attiecīgos elementus a, b un u, (4-to elementu no zīmju matricām ($s_{k\,m}$), 2-o elementu no *sin/cos* matricām un 4-o elementu no pakāpju zīmju matricām), iegūst:

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

Visus struktūru nosakošos parametrus (a, b un u) varam apvienot vienā, kopējā indeksā (*id3*) [P11]:

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

Struktūras parametrus no *id3* var iegūt izmantojot vienkāršas sakarības:

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

kur fix un $round$ – atbilstoši noapaļošanas veidi (detalizētāk 3.2.1. apakšnodaļā).

Indeksu *id3* dažkārt ir grūti izmantot automatizētos algoritmos, jo, piemēram, pēc 114 kā nākamais seko 121. Tāpēc tiek ieviests indekss $id1 \in [1, 64]\,, id1 \in \mathbb{N}$, kuru aprēķina sekojoši:

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

1. tabulā attēlotas dažas no iespējamajām 64 Jakobi rotācijas matricas struktūrām ar visiem no aprakstītajiem universālās matricas indeksēšanas variantiem. [3] un [14] izman-

1. tabula. Elementāro vispārināto rotācijas matricu piemēri [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	$\begin{bmatrix} +\cos(\phi)\cdot e^{+j\psi} & +\sin(\phi)\cdot e^{-j\gamma} \\ +\sin(\phi)\cdot e^{+j\gamma} & -\cos(\phi)\cdot e^{-j\psi} \end{bmatrix}$

totas reālās rotācijas matricas (piemēram, Givensa rotācijas matrica). Jāpatur prātā, ka reālu rotācijas matricu var uzskatīt par kompleksās matricas vienkāršotu gadījumu (1.10). Reālie ortogonālie pārveidojumi un to ierīces apskatītas [P1]-[P7].

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

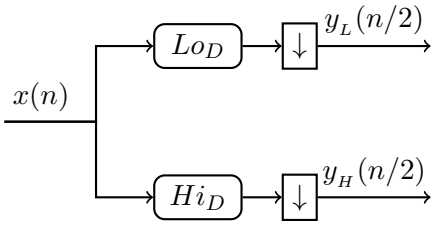
Komplekso vektoru, kas iegūts pagriežot jebkuru kompleksu ieejas vektoru $\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$ par izvēlētajiem leņķiem ϕ, ψ un γ, var uzskatīt par divu nolašu signāla spektru $\begin{bmatrix} y_1 \\ y_2 \end{bmatrix}$ ϕ, ψ un γ atbilstošā bāzē:

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

Izvēloties divus FIR filtrus ar attiecīgiem koeficientiem $b_L = [\cos(\phi)\cdot e^{i\cdot\gamma}, \sin(\phi)\cdot e^{-i\cdot\psi}]$ un $b_H = [-\sin(\phi)\cdot e^{i\cdot\psi}, \cos(\phi)\cdot e^{-i\cdot\gamma}]$ (kā arī veicot decimāciju filtru izejās), pēc divu ieejas nolašu apstrādes, varam iegūt to pašu signāla spektru (skat. 1.1. att.).

Ortogonālie filtri un to īpašības ir sīkāk apskatītas 2.4. nodaļā. Lai atjaunotu signālu, iegūtais spektrs jāreizina ar inverso $\mathbf{T}^{-1}$ matricu (kompleksas matricas gadījumā - ar Ermita matricu) [P9]:

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



1.1. att. Divu nolašu dekompozīcija

Ģeometriski signāla atjaunošana nozīmē kompleksā vektora pagriešanu pretējā virzienā.

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

1.2.2. Pilnā unitārā matrica

Reāli izmantojami diskrēti signāli parasti ir garāki par 2 nolasēm. Turpmāk aplūkosim signālus, kuru garums ir $N = 2^n$ nolases. N−dimensiju telpā mēs viegli varam veikt vienlaicīgu N−dimensiju vektora rotāciju $N/2$ neatkarīgās plaknēs. Matricu, kura veic minētās rotācijas var konstruēt dažādi. Viens no veidiem ir kāpņveida faktorizētās matricas izmantošana. Izmantojot $\mathbf{T}_U(\phi,\ \psi,\ \gamma,\ 424)$ struktūru, iegūstam sekojošu 4×4 unitāru kāpņveida matricu:

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

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

No pieraksta viedokļa daudz ērtāk ir izmantot blokveida struktūru. Šādi pierakstot matricu, labāk ir redzamas neatkarīgās rotācijas struktūras:

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

Kāpņveida matricai ir būtiska priekšrocība (kas tiek izmantota jauna veida ortogonālo pārveidojumu iegūšanai) salīdzinot to ar blokveida matricu – sareizinot $\log_2(N)$ matricas ar kāpņveida struktūru, tiek iegūta "pilna" (vispārīgā gadījumā - nulles nesaturoša) matrica. Tā, piemēram, 4×4 Adamāra (*Hadamard*) matricu var iegūt no 2 vienādām kāpņveida matricām (izmanto Adamāra ātrajam pārveidojumam [16]):

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

Kāpņveida matricu no blokveida matricas (un otrādi) iegūst pārkārtojot matricas rindas, jeb formāli izmantojot konkrētam gadījumam atbilstošu permutāciju matricu:

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

kur

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

Līdzīgi kā (1.16), no 2 kāpņveida matricām iegūstam "pilnu", 4×4 unitāru pārveidojuma matricu (1.19). Vispārīgā gadījumā, katrā no rotācijām iespējams izmantot savu leņķu komplektu. Lai nedaudz atvieglotu pierakstu, pieņemam sekojošus apzīmējumus – $\Phi_k = (\phi_k, \psi_k, \gamma_k)$.

$$\mathbf{H}_4 = \mathbf{B}_s\left(\Phi_{11}, \Phi_{12}\right) \cdot \mathbf{B}_s\left(\Phi_{21}, \Phi_{22}\right) =$$

$$= \begin{pmatrix} c_{11}\, c_{21}\, e^{\psi_{11}\, i}\, e^{\psi_{21}\, i} & c_{11}\, s_{21}\, e^{-\gamma_{21}\, i}\, e^{\psi_{11}\, i} & c_{22}\, s_{11}\, e^{-\gamma_{11}\, i}\, e^{\psi_{22}\, i} & s_{11}\, s_{22}\, e^{-\gamma_{11}\, i}\, e^{-\gamma_{22}\, i} \\ -c_{12}\, s_{21}\, e^{\gamma_{21}\, i}\, e^{\psi_{12}\, i} & c_{12}\, c_{21}\, e^{\psi_{12}\, i}\, e^{-\psi_{21}\, i} & -s_{12}\, s_{22}\, e^{-\gamma_{12}\, i}\, e^{\gamma_{22}\, i} & c_{22}\, s_{12}\, e^{-\gamma_{12}\, i}\, e^{-\psi_{22}\, i} \\ -c_{21}\, s_{11}\, e^{\gamma_{11}\, i}\, e^{\psi_{21}\, i} & -s_{11}\, s_{21}\, e^{\gamma_{11}\, i}\, e^{-\gamma_{21}\, i} & c_{11}\, c_{22}\, e^{-\psi_{11}\, i}\, e^{\psi_{22}\, i} & c_{11}\, s_{22}\, e^{-\gamma_{22}\, i}\, e^{-\psi_{11}\, i} \\ s_{12}\, s_{21}\, e^{\gamma_{12}\, i}\, e^{\gamma_{21}\, i} & -c_{21}\, s_{12}\, e^{\gamma_{12}\, i}\, e^{-\psi_{21}\, i} & -c_{12}\, s_{22}\, e^{\gamma_{22}\, i}\, e^{-\psi_{12}\, i} & c_{12}\, c_{22}\, e^{-\psi_{12}\, i}\, e^{-\psi_{22}\, i} \end{pmatrix} \tag{1.19}$$

kur

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

Iegūtā kompleksā matrica $\mathbf{H}_4$ pie visām parametru vērtībām ir unitāra. Par to var pārliecināties, veicot kompleksas matricas rindu skalāro reizinājumu:

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

kur H_n – n-tā matricas H rinda.

Līdzīgā veidā var iegūt arī kompleksās ortonormālās matricas ar lielāku dimensiju skaitu. Matricas izmēram ir jābūt

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

Analītiskais pieraksts (1.19) pat pie $N = 4$ ir sarežģīts un grūti izprotams. Tas parādīts, lai ilustrētu vispārinātā uz leņķiem bāzētā kompleksā ortogonālā pārveidojuma (**CRABOT** – ***C**omplex **R**otation **A**ngle **B**ased **O**rthogonal **T**ransform*) matricas iegūšanu. Matricas iegūšanai nepieciešamo parametru skaits ir $n_{par} = 3 \cdot \frac{N}{2} \cdot \log_2(N)$. Apkoposim visus iespējamos parametrus vienā leņķu/parametru trīs slāņu matricā (1.23) [P1], [P5], [P8], [P10], [P11].

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

Ja $\mathbf{\Phi}_p$ ir parametru matricas $\mathbf{\Phi}$ p−tā kolonna, tad gadījumā, kad

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

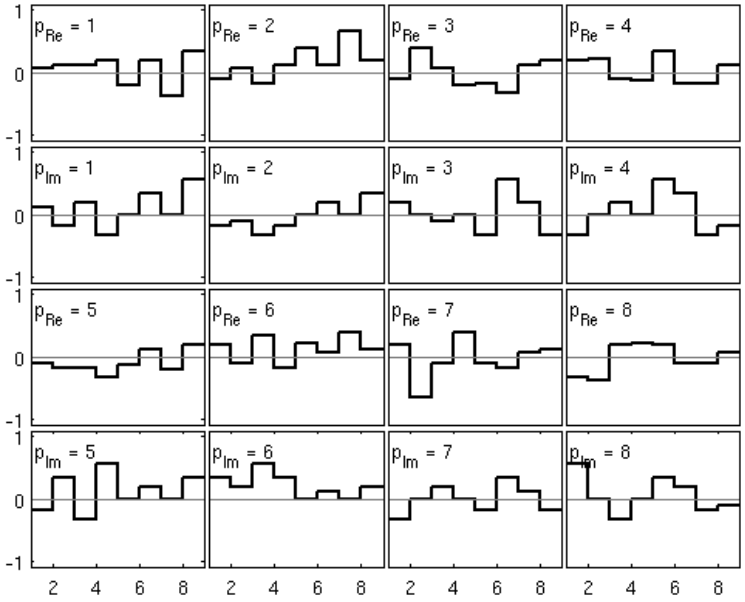
kur $\mathbf{\Phi}_p$ – parametru matricas p−tā kolonna, Φ_p – parametru kopums,

tiek iegūts viens no uz leņķiem bāzēta pārveidojuma vienkāršotajiem gadījumiem – **CCRAIMOT** (***C**omplex **C**onstant **A**ngle **I**n **M**atrix **O**rthogonal **T**ransform*). Šī pārveidojuma klase joprojām ļauj iegūt bezgalīgi lielu (bet ne ar jebkuras formas BF) ortogonālu matricu skaitu, un ir vadāma ar $n_{Par} = 3 \cdot \log_2 N$ parametriem.

CCRAIMOT matricas iegūšanas piemērs, kad $N = 8$:

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

Bāzes funkcijas (1.25) aprēķinātajam gadījumam attēlotas 1.2. attēlā.



1.2. att. **CCRAIMOT** bāzes funkcijas, $N = 8$,
$\phi = [60°, 60°, 60°]$, $\psi = [60°, -60°, -60°]$, $\gamma = [60°, 60°, -60°]$

Ja $N = 2^n, n > 2, n \in \mathbb{N}$, tad (1.25) varam vispārināt sekojošā veidā:

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

N nolases gara signāla dekompozīciju matricu formā var veikt, izmantojot ātro [17],[18] algoritmu – reizinot ieejas signālu ar $\log_2(N)$ kāpņveida matricām (1.27).

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

Ātrā un klasiskā algoritma salīdzinājums pēc aritmētisko darbību skaita parādīts 2. tabulā. Tabulā attēlots reizinājumu (M) un summēšanu (A) skaits. Reizinājumu un summēšanu skaitu, kas nepieciešams, lai sareizinātu $N \times N$ kompleksu matricu ar N nolases garu kompleksu vektoru, aprēķina izmantojot sekojošas formulas:

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

2. tabula. Ātrā un klasiskā algoritma salīdzinājums.
Aritmētisko operāciju skaits

N	**Ātrais**		**Klasiskais**	
	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

Izmantojot ātro algoritmu, aritmētisko operatoru skaits ir:

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

1.2.3. Hārveidīgie kompleksie ortogonālie pārveidojumi

Šajā apakšnodaļā apskatītie Hārveidīgie kompleksie ortogonālie pārveidojumi ir iekļauti izveidotajā *MatLab* rīkkopā, kas publicēta [P8] (2. tabula). Šo pārveidojumu veidošanas metodika ir līdzīga reālo vispārināto (parametrisko) Hārveidīgo pārveidojumu veidošanas metodikai, kas ir detalizēti aprakstīta [15]. Tā kā ierobežotā publikācijas apjoma dēļ, [P8] un arī [15] nav parādītas kompleksā pārveidojuma izvērstākas izteiksmes, veiksim to īsu apskatu šeit.

Izmantojot elementārās kompleksās rotācijas matricas, varam iegūt arī kompleksos parametriskos Hāra pārveidojumus. Robežgadījumā, kad $\phi = \pi/4$, $\psi = 0$, $\gamma = 0$, elementārā kompleksā rotācijas matrica top par klasisko elementāro Hāra matricu:

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

Lielāka izmēra Hāra matricai ir būtiska atšķirība no 1.2.2. nodaļā apskatītajām matricām – visas Hāra funkcijas (matricas rindas), izņemot pirmās divas, ir ar lokalizētu enerģiju, t.i., blakus esošiem matricas elementiem lokālā intervālā ir nenulles vērtība. Parametrisko Hāra matricu, līdzīgi kā 1.2.2. nodaļā apskatītajos pārveidojumos, iegūst no $\log_2(N)$ faktorizētajām matricām. Hāra pārveidojumiem, atšķirībā no iepriekšējiem, mainoties matricas izmēriem, mainās faktorizēto matricu struktūra un parametru skaits tajās. Tā, piemēram, vispārināto Hārveidīgo matricu $N = 4$ gadījumam, iegūst sekojoši:

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

(1.31) izteiksmē, kreisā faktorizētā matrica sastāv no elementārās unitārās rotācijas matricas, nullēm un vienības matricas. Atsevišķas kompleksās rotācijas matricas struktūras, pie ieejas parametriem ($\phi = 0, \psi = 0, \gamma = 0$) ir vienādas ar vienības matricu. Lai padarītu ērtāku Hārveidīgo matricu pierakstu, turpmāk, ja $\Phi_k = 0$, izvēlēsimies tās elementārās rotācijas matricas struktūras, kurām spēkā ir sekojoša sakarība:

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

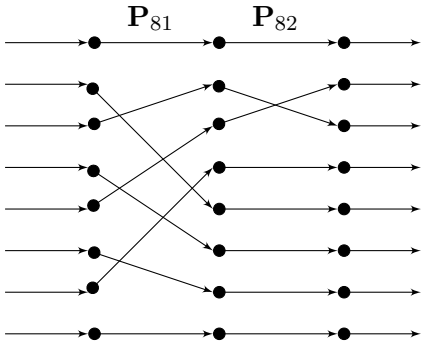
Tādā gadījumā, zinot faktorizēto matricu struktūru, Hārveidīgā pārveidojuma matricu, kad $N = 4$ (1.31), varam izteikt ar sekojošu parametru matricu:

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

Vispārīgā gadījumā, ja $N = 2^n, n > 2, n \in \mathbb{N}$, ievērojot nosacījumu (1.32), Hārveidīgā pārveidojuma matricu varam aprakstīt ar parametru matricu [P3], [P5], [P8] tā, ka katrā nākamajā parametru matricas kolonnā ir divas reizes mazāk nenulles elementu nekā iepriekšējā:

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

Līdzīgi kā 1.2.2. nodaļā, arī Hārveidīgos pārveidojumus var sadalīt apakšklasēs. Tā, piemēram, ņemot tikai vienu parametru komplektu Φ_k katrā no faktorizētajām matricām (līdzīgi kā 1.2.2. nodaļā aprakstītajā **CCRAIMOT** klasē), iegūstam **CRAIM-HT** – ***C**onstant **R**otation **A**ngle **I**n **M**atrix - **H**aar **T**ransform.* Izmantojot permutāciju matricu $\mathbf{P}_4$ (1.18), universālo kompleksās matricas pieraksta formu $\mathbf{T}_U$, nuļļu matricu $\mathbf{0}$ un vienības



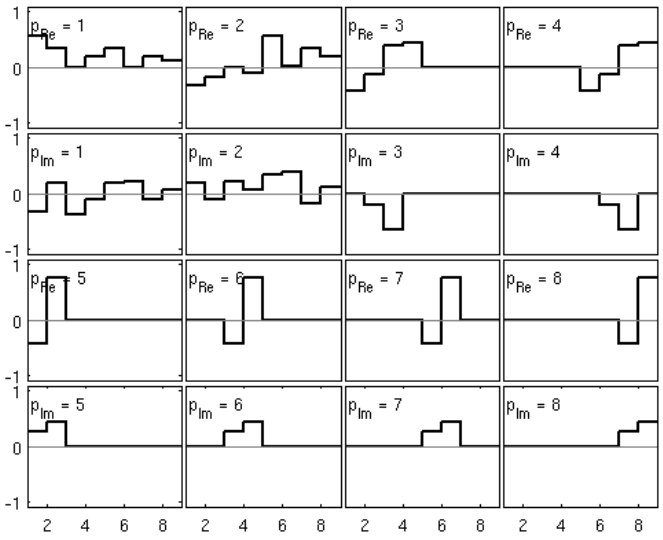
1.3. att. Hārveidīgo pārveidojumu permutāciju matricu grafisks attēlojums ($N = 8$)

matricu $\mathbf{I}$, 4−nolašu **CRAIM-HT** pārveidojumu varam pierakstīt sekojoši:

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

kur $\mathbf{\Phi}_H$ – parametru matrica (1.33), indekss pie $\mathbf{I}$ un $\mathbf{0}$ norāda, attiecīgi, vienības un nuļļu matricas izmēru.

Lai iegūtu Hārveidīgo pārveidojuma matricu $N = 8$ gadījumam, jāzina faktorizēto matricu struktūra, kuru nosaka permutāciju matricas (to formēšanas algoritmu var atrast [15]), un atbilstošā parametru matrica. Permutāciju matricu lietošana atvieglo sakarību pierakstu, jo faktorizētās matricas ļauj attēlot ar komplekso rotācijas matricu $\mathbf{T}_U$ palīdzību. Grafisks permutāciju matricu attēlojums (1.3. att.) palīdz labāk izprast to veidošanu, kas ir būtisks nosacījums lielāku izmēru ($N = 2^n$, kur $n > 2$ un $n \in \mathbb{N}$) Hārveidīgo



1.4. att. **CCRA-HT** bāzes funkcijas, $N = 8$,
$\Phi_1 = (30°, -30°, -30°)$, $\Phi_2 = (30°, 30°, 30°)$, $\Phi_3 = (30°, -30°, -30°)$

pārveidojumu iegūšanā.

Izmantojot 1.3. attēlā attēlotās permutāciju matricas, sekojošā veidā iegūstam **CCRAIM-HT** pārveidojuma matricu:

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

1.4. attēlā parādītas visas **CCRAIM-HT** bāzes funkcijas, ja parametru matrica ir:

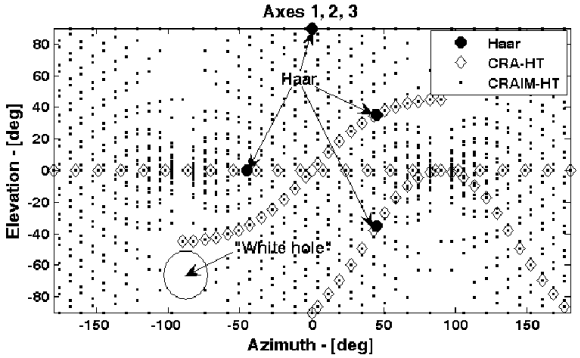
$$\mathbf{\Phi}_H=\begin{bmatrix}\Phi_1 & \Phi_2 & \Phi_3\\ \Phi_1 & \Phi_2 & 0\\ \Phi_1 & 0 & 0\\ \Phi_1 & 0 & 0\end{bmatrix}=\begin{bmatrix}(30°,-30°,-30°) & (30°,30°,30°) & (30°,-30°,-30°)\\ (30°,-30°,-30°) & (30°,30°,30°) & (0,0,0)\\ (30°,-30°,-30°) & (0,0,0) & (0,0,0)\\ (30°,-30°,-30°) & (0,0,0)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

Hārveidīgo pārveidojumu apakšklases, atkarībā no parametru matricas $\mathbf{\Phi}_H$ nenulles elementu ierobežojumiem, parādītas 3. tabulā. Hārveidīgo pārveidojumu apraksts [P8] ir līdzīgs reālo Hārveidīgo pārveidojumu aprakstam [P3], [P5].

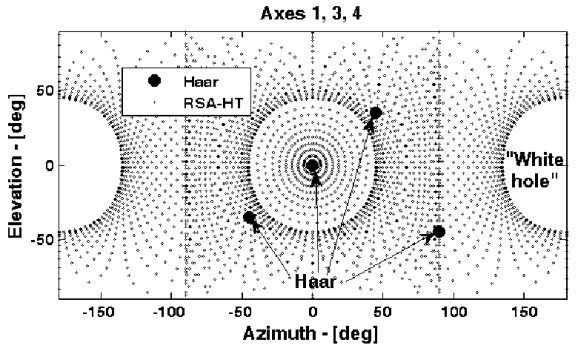
Tā kā 4−D telpa (kā arī visas daudzdimensiju telpas pie $N>4$) ir grūti iedomājama,

3. tabula. **CCRA-HT** pārveidojumu apakšklases [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**

1.5. att. 4−dimensionāla vektora galu projekciju 3−dimensionālā telpā izklājums 2−D plaknē [P5]

tad 4−dimensionālu BF-u jebkuras trīs nolases mēs varam attēlot kā projekcijas 3−D telpā (pavisam iespējamas $C_4^3 = 4$ 4−dimensionāla signāla 3−D projekcijas). Atzīmējot vektoru galu projekcijas uz 3−D sfēras un šo sfēru pēc tam izklājot plaknē, iegūstam BF-u vektoru galu projekciju pārklājumus (skat. 1.5. att.). Pilnībā noklātas sfēras visām projekcijām liecinātu, ka ar izvēlēto pārveidojuma klasi, mainot leņķus iespējams iegūt jebkuras formas BF-as. Kā redzams, kopējā tendence ir tāda, ka, palielinot leņķu (parametru) skaitu, noklājums kļūst arvien blīvāks.

Rezumējums

Iegūtie rezultāti

- Ieviests vispārinātais elementārās unitārās rotācijas matricas pieraksts un uz tā pamata iegūtas visas iespējamās Jakobi matricas vispārinājumu struktūras (ja ϕ,ψ,$\gamma \in$ [0,45°], tad iespējamas 64 struktūras).

Secinājumi

- Mainot parametrus ϕ,ψ un γ, varam iegūt bezgalīgi daudz dažādu unitāru fī-pārveidojumu.

- Modificējot faktorizētās matricas tā, ka daļa rotācijas leņķu tiek izvēlēti vienādi ar nulli, iespējams iegūt impulsveida *BF* (vispārinātās parametriskās Hārveidīgās funkcijas, kas ir līdzīgas veivletiem (*Wavelets*)).

- Ātro algoritmu lietošana būtiski samazina pārveidojuma aritmētisko operāciju skaitu 1.2.2. nodaļā aprakstītajiem pārveidojumiem, ja $N > 4$, bet Hārveidīgajiem pārveidojumiem, ja $N > 8$.

2. Ortogonālo pārveidojumu realizācijas arhitektūras

Šajā nodaļā apskatīsim iespējamo fī-pārveidojumu realizāciju arhitektūru teorētisko pusi[1]:

1. Pārveidojuma realizācijas arhitektūras paralēlā struktūra,
2. Pārveidojuma realizācijas arhitektūras virknes struktūra,
3. Signāla dekompozīcijas-rekonstrukcijas kokveida struktūra,
4. Signāla dekompozīcija, izmantojot kompleksus FIR filtrus.

2.1. Pārveidojuma realizācijas arhitektūras paralēlā struktūra

Tiešā veidā realizējot sakarību (1.27), iegūstam pārveidojuma paralēlo struktūru. Pārveidojums tiek veikts izpildot sekojošus soļus:

1. Tiek savākts N nolases liels datu bloks,
2. Savāktās nolases reizina ar $\log_2(N)$ retinātajām kāpņveida rotācijas matricām $\mathbf{B}_s$,
3. Tiek izvadīti rezultāti un paralēli vāktas nākamā datu bloka nolases.

Par pārveidojuma realizācijas arhitektūras paralēlo struktūru var uzskatīt arī [P5] apskatīto analizatoru-sintezatoru, jo pārveidojums tiek veikts līdzīgā veidā kā iepriekš pieminētais. Tikai, atšķirībā no augstāk šajā apakšnodaļā minētā, savākto nolašu pāri, kuri tiek uzskatīti par divdimensionāliem signālu vektoriem, tiek rotēti ar vairākiem paralēliem *CORDIC RE* (skat. 3.3. nodaļu).

2.2. Pārveidojuma realizācijas arhitektūras virknes struktūra

Ar pārveidojuma realizācijas arhitektūras virknes struktūru jāsaprot tāda struktūra, kurā tiešā veidā netiek veikts pārveidojums, bet, izmantojot zemāk aprakstīto algoritmu, tiek iegūta jebkura fī-pārveidojuma matricas $\mathbf{H}$ vērtība.

2.2.1. CRAIMOT bāzes funkciju ģenerators [P1], [P4]

Apskatīsim reālu **CRAIMOT** *BF* ģenerēšanu un salīdzināsim pēc aritmētisko darbību skaita divus ģeneratora variantus:

1. **CRAIMOT** pārveidojuma $\mathbf{H}$ matricas iegūšana ar ātro algoritmu,
2. *BF* nolašu iegūšana ar *sin/cos* reizinājumiem.

[1] Realizētās eskperimentālās iekārtas sīkāk apskatītas kopsavilkuma 4. nodaļā

Nepieciešamo aritmētisko darbību skaitu, reālas pārveidojuma matricas aprēķinam ar ātro algoritmu, var noteikt izmantojot sekojošas sakarības [P4]:

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

Visu *BF* saglabāšanai nepieciešami arī ievērojami atmiņas resursi (N^2 atmiņas šūnas). Izmantojot *sin/cos* algoritmu, K *BF*-u iegūšanai nepieciešamo aritmētisko operāciju skaitu nosaka sekojoši [P4]:

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

Virknes arhitektūras būtība ir jebkuras *BF* nolases iegūšana ar *sin/cos* reizinājumiem– $p-$tās *BF*-as $t-$to vērtību iegūst, izmantojot sekojošu sakarību [P1], [P4], [3]:

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

Pakāpju vērtības (a_j, m_j un k_j) tiek noteiktas atkarībā no elementārās rotācijas matricas struktūras (skat. 4. tabulu).

4. tabula. Pakāpju aprēķina izteiksmes (ņemtas no [3], papildinātas)

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

Tā, piemēram, izvēloties leņķu vērtības $\phi = [45°,\ 30°,\ 60°,\ 60°]$ un $\mathbf{T}_{R+}$ rotācijas matricas struktūru, $6-$ās ($p = 6$) *BF* $13-$o vērtību ($t = 13$) iegūstam izmantojot (2.3) un 5. tabulas datus:

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

kur $\{...\}_2$ – indeksa binārais attēlojums.

Visu *BF*-u iegūšanai izmantojot aprakstīto algoritmu, nepieciešams lielāks aritmētisko operāciju skaits nekā izmantojo ātro transformāciju (*FT*). Mēs varam izteikt šī algoritma aritmētisko darbību efektivitātes koeficientu, salīdzinot to ar *FT* [P4]:

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

5. tabula. Koeficientu aprēķins piemēram (2.4) [P1]

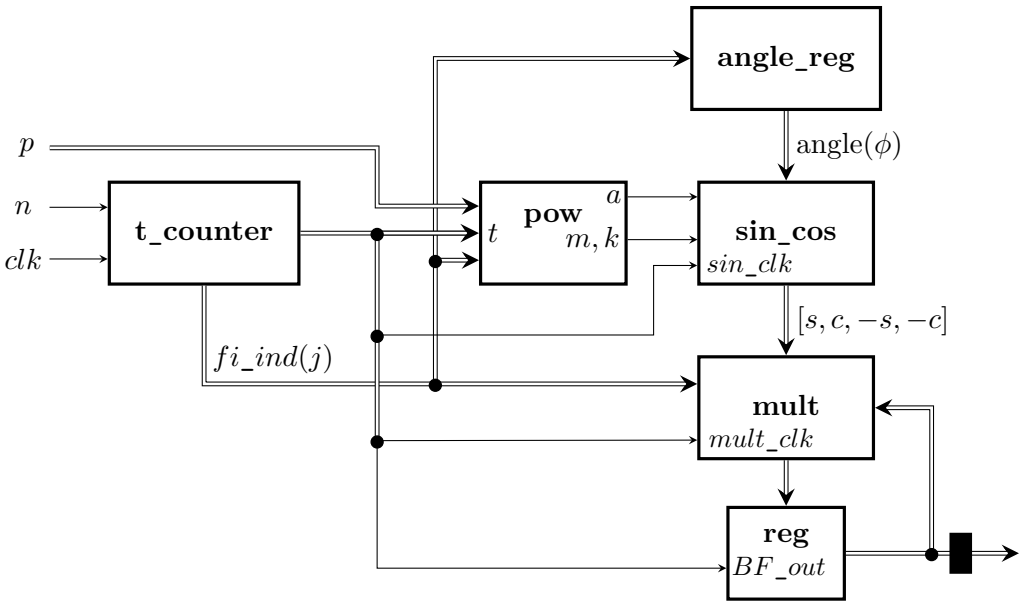
j-tais pakāpes koeficients	**Binārā izteiksme**	**Vērtība**
a_1 (LSB)	$1 \cdot 0 = 0$	$(-1)^0 = 1$
m_1 (LSB)	$1 \cdot 0 + 0 \cdot 1 = 0$	$(s_1)^0 = 1$
k_1 (LSB)	$0 \cdot 0 + 1 \cdot 1 = 1$	$(c_1)^1 = c_1$
a_2	$0 \cdot 0 = 0$	$(-1)^0 = 1$
m_2	$0 \cdot 0 + 1 \cdot 1 = 1$	$(s_2)^1 = s_2$
k_2	$1 \cdot 0 + 0 \cdot 1 = 0$	$(c_2)^0 = 1$
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)	$1 \cdot 1 + 0 \cdot 0 = 1$	$(c_4)^1 = c_4$

Gadījumā, ja

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

aritmētisko operāciju skaits abiem *BF*-u iegūšanas variantiem ir vienāds. Izmantojot (2.1), (2.2) un (2.6), mēs varam iegūt maksimālo *BF*-u skaitu (K_{max}), kuru atsevišķa ģenerēšana izmantojot apskatīto algoritmu ir tikpat efektīva kā FT izmantošana [P4]:

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



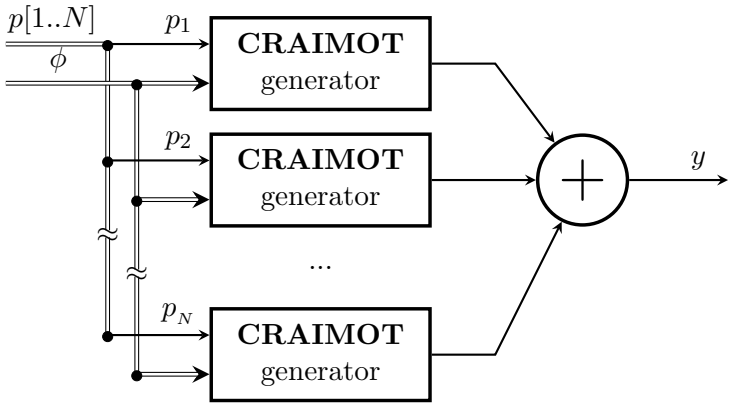
2.1. att. Vienkāršota *BF* ģeneratora blokshēma [P1]

Praktiskos pielietojumos ērtāk ir izmantot sekojošu sakarību [P4]:

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

kur $floor$ – noapaļošanas veids (sk. 3.2.1).

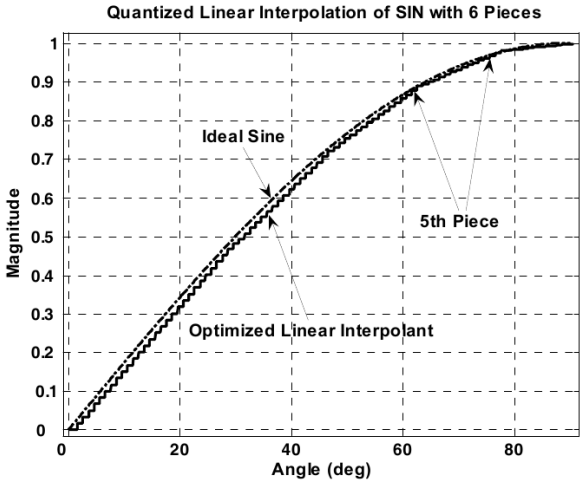
Izmantojot (2.3) un 4. tabulas datus, varam izveidot *BF*-u ģeneratoru (skat. 2.1. att.), kurā katra *BF* vērtība tiek katrreiz aprēķināta. Tas nodrošina vienmērīgu sistēmas darbību un niecīgu atmiņas resursu nepieciešamību, kā arī ļauj ģenerēt ļoti garas ($N = 2^{32}$) *BF*-as. Uz šāda ģeneratora bāzes var izveidot vienkāršu signālu sintezatoru (skat. 2.2. att.).



2.2. att. Vienkāršota signālu sintezatora blokshēma

***BF*-u ģeneratora kļūdu novērtējums** Apskatītā ģeneratora kļūdu avots ir *sin/cos* vērtību aprēķināšanas metode. Māris Tērauds (*MT*) veica kļūdu aprēķinu *BF*-ām un parciālu *BF*-u summām, ja *sin/cos* vērtības aprēķinātas ar divām metodēm – lineārā interpolācija (skat. 2.3. att.) un *CORDIC*. Eksperimentāli tika pārbaudīts, ka reāla uz *FPGA* izveidota ģeneratora *BF*-u un *BF*-u parciālo summu vidējās kvadrātiskās kļūdas (*MSE*) iekļaujas *MT* teorētiski aprēķināto kļūdu robežās [3],[P7].

Jaunākajiem *FPGA* arvien palielinās iekšējās atmiņas apjomi, tāpēc arvien efektīvāk ir iespējams izmantot *sin/cos* vērtību tabulu, kas ir visātrākā no *sin/cos* vērtību iegūšanas metodēm. Īsumā apskatīsim tabulas *sin/cos* vērtību iegūšanas kļūdas, jo šo kļūdu novērtējums neparādās publicētajos rakstos un *MT* disertācijā [3]. *sin/cos* tabulā, visas leņķu vērtības $\alpha \in [0,\ 90^\circ]$ (visas pārējās sinusa vai kosinusa vērtības var elementāri noteikt zinot $\sin(\alpha)$ uzrādītajam intervālam) tiek vienmērīgi sadalītas 2^{n_α} daļās, kur n_α - argumenta bitu skaits. Sinusa vērtības tabulā tiek saglabātas $Q1.x$ fiksētā punkta formātā (x - *FPA WL*). No minētā varam secināt, ka tabulas *sin/cos* kļūda sastāv no divām komponentēm– leņķa kļūda un *FPA* kļūda. Lielāko leņķa kļūdu (α_{err}) iegūsim tad, ja izvēlētais leņķis būs tieši pa vidu tabulā definētajiem. Zinot, ka sinusa funkcija visjūtīgākā pret leņķa izmaiņām ir tad, kad leņķa vērtība ir tuvu nullei, varam noteikt maksimālo tabulas sinusa leņķa



2.3. att. Sinusa funkcijas optimizēta lineārā interpolācija, $\Delta_{max} \approx 0.004$ [P4]

kļūdu, ko normējam pret tabulā definētā sinusa vērtību diapazonu (ja $\alpha \in [0,\, 90°]$, tad $\sin(\alpha) \in [0, 1]$):

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

FPA kļūdas lielums atkarīgs no sinusa vērtību vārda garuma (*WL*) un noapaļošanas veida (skat. 3.5. nodaļu). Parasti, sagatavojot sinusa tabulas, izmanto visprecīzāko (*round*) noapaļošanu. Sinusa tabulai nepieciešamo atmiņas daudzumu bitos aprēķina, izmantojot sekojošu sakarību:

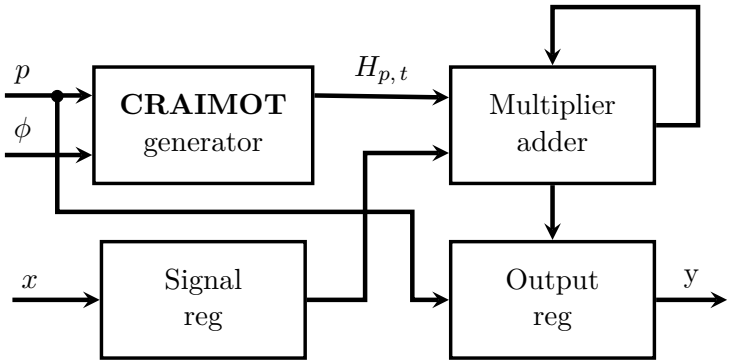
$$N_{bits} = 2^{n_\alpha} \cdot x, \tag{2.10}$$

kur 2^{n_α} - tabulas vērtību skaits, x - $Q1.x$ *FPA* aritmētikas *WL*.

2.2.2. Signālu spektra analizators CRAIMOT funkciju bāzē [P2]

Vienkāršu signālu analizatoru (skat. 2.4. att.) varam izveidot par pamatu ņemot *BF*-u ģeneratoru. Šādā veidā realizēts signālu analizators ļauj mums aprēķināt konkrētus spektra koeficientus, izmantojot signāla un attiecīgās *BF* skalāro reizinājumu. Visu spektra koeficientu aprēķināšana prasa zināmu laiku, jo secīgi pienākošās ieejas signāla nolases tiek saglabātas signālu reģistrā un pēc tam N reizes tiek veikts vektoru skalārais reizinājums (signāla vektors skalāri tiek pareizināts ar visām *BF*-ām).

Šajā nodaļā apskatītajā signālu analizatorā spektra iegūšanas darbības tiek izvērstas laikā, tāpēc tā ātrdarbības parametri ir salīdzinoši zemi. 2.3. nodaļā aplūkots kokveida spektra iegūšanas algoritms, kas ir daudz ātrdarbīgāks, jo visi spektra koeficienti tiek iegūti paralēli.



2.4. att. Vienkāršota signālu analizatora blokshēma [P2]

Rezumējums par publikācijām [P1], [P2], [P4]

Iegūtie rezultāti

- Izveidota uz *sin/cos* reizinājumiem balstīta **CRAIMOT** ģeneratora virknes arhitektūra un tam atbilstoša signālu sintezatora arhitektūra.
- Izveidota **CRAIMOT** spektra analizatora virknes arhitektūra.
- Apskatītas trīs dažādu algoritmu alternatīvas *sin/cos* vērtību iegūšanai.

Secinājumi

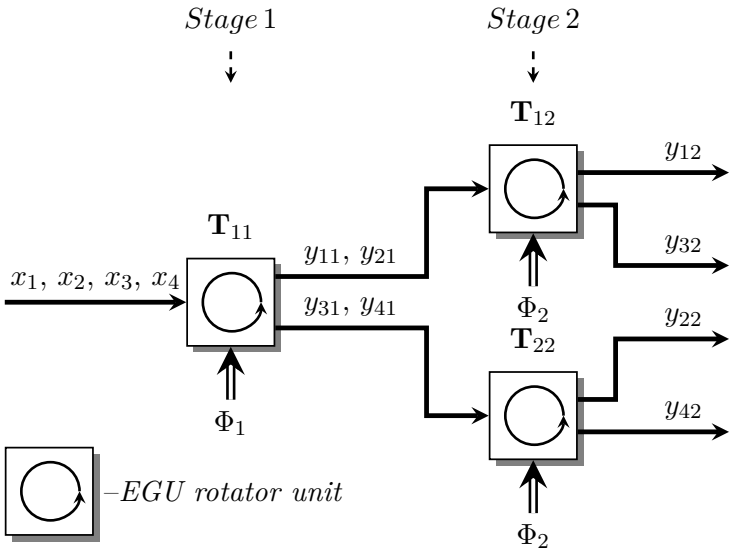
- Dažu (2.7),(2.8) *BF* ģenerēšana ar *sin/cos* reizinājumiem ir efektīvāka par *BF* iegūšanu ar ātro algoritmu, tā, piemēram, ja $N = 64$, tad, izmantojot virknes algoritmu, mazāk nekā 9 *BF* iegūšanai nepieciešamas mazāk aritmētiskās darbības, nekā izmantojot ātro pārveidojumu.
- *Sin/cos* vērtību iegūšanas metode nosaka gan ģenerēto *BF* nolašu kļūdu, gan izmantojamo resursu (realizējot *FPGA–LE*, atmiņas biti, *DSP* elementi) skaitu.
- Uz *BF* ģeneratora balstīts signālu spektra analizators ir salīdzinoši viegli izveidojams, bet tam ir zema ātrdarbība, jo katras spektra vērtības iegūšanai ir secīgi jāveic N nolašu garu vektoru skalārais reizinājums.

2.3. Signāla dekompozīcijas-rekonstrukcijas sistēmas kokveida struktūra

Kokveida struktūra pārveidojumus ļauj izveidot izmantojot konveijera (*pipeline*) arhitektūru [19], kas efektīvāk izmanto skaitļošanas (šajā gadījumā *FPGA*) resursus.

2.3.1. Signāla dekompozīcija [P10],[P11]

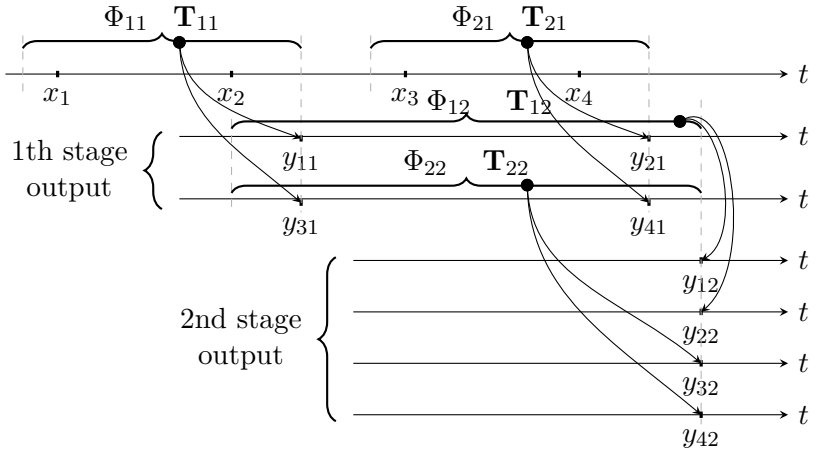
Šajā nodaļā apskatītas likumsakarības starp matricu formā un ar kokveida struktūru realizētu signāla dekompozīciju. 2.5. attēlā parādīta signāla dekompozīcijas kokveida



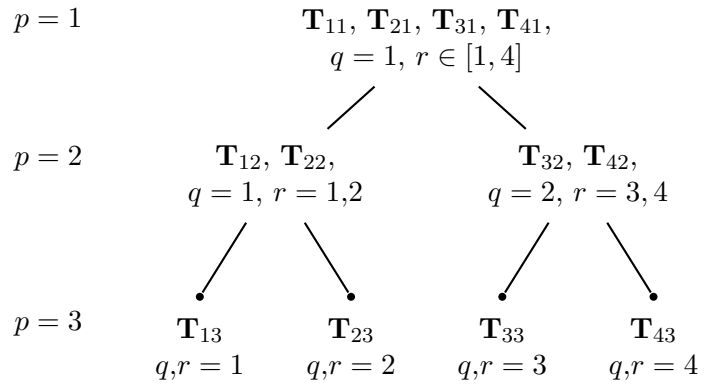
2.5. att. Vienkāršota **CCRAIMOT** dekompozīcijas blokshēma, $N = 4$ [P11], [P10]

struktūras ar rotācijas elementiem (*EGU rotator unit*) vienkāršota blokshēma. Rotācijas elementu parametri var laikā mainīties, tādējādi iegūstot **CRABOT** klases spektru. Parametru maiņas laika diagramma $N = 4$ gadījumam parādīta 2.6. attēlā. Savukārt 2.7. attēlā parādīta shematiska parametru maiņa $N = 8$ gadījumam. Lai arī katrai rotācijas matricas struktūrai ir savs atbilstošais rotācijas elements, izmantojot 3.6. nodaļā aprakstīto automatizēto algoritmu, šo elementu iegūšana nesagādā grūtības.

4 nolašu signāla spektru matricu formā, izmantojot ātro **CCRAIMOT** pārveidojumu (divas vienādas struktūras kāpņveida matricas, kur katrā no matricām ir vieni un tie paši rotācijas parametri), iegūstam sekojoši:



2.6. att. Vienkāršota laika diagramma [P11]



2.7. att. Dekompozīcijas kokveida struktūras un adresācijas piemērs, $N = 8$ [P11]

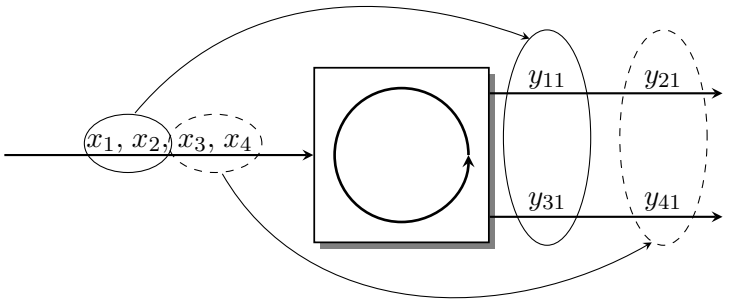
$$\mathbf{Y}_4 = \mathbf{B}_s\left(\Phi_2\right) \cdot \mathbf{B}_s\left(\Phi_1\right) \cdot \mathbf{X}_4 \tag{2.11}$$

Izvērstā veidā pierakstot signāla un pirmās matricas reizinājumu, iegūstam:

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

Signāla dekompozīciju realizējot konveijera (*pipeline*) arhitektūrā, 4−nolašu signāla dekompozīcijai nepieciešamas divas kaskādes. Pirmā no tām parādīta 2.8. attēlā.

Tā kā nolases rotācijas elementa (pagriezēja) ieejā nonāk secīgi pēc kārtas, tad tādā



2.8. att. Četru nolašu dekompozīcijas pirmā kaskāde

pašā secībā tās arī pa pāriem tiek apstrādātas. No virknē pēc kārtas ienākošām 2 nolasēm tiek iegūti 2 paralēli starprezultāti ar divreiz zemāku frekvenci. Matemātiski to var pierakstīt kā divu neatkarīgu 2×2 ($\mathbf{T}_U$) rotācijas matricu reizinājumu ar vektoriem $\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$ un $\begin{bmatrix} x_3 \\ x_4 \end{bmatrix}$, vai arī kā 4 nolases gara signāla reizinājumu ar blokveida rotācijas matricu (1.15):

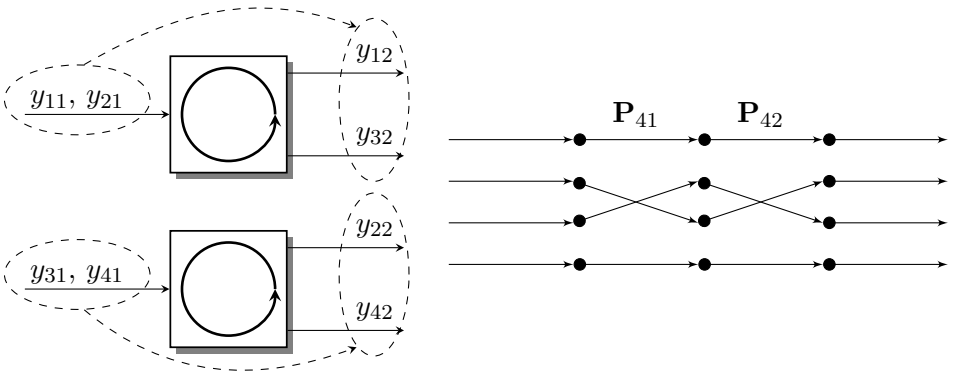
$$\mathbf{Y}_{b1} = \mathbf{B}_b(\Phi_1) \cdot \mathbf{X} = \begin{bmatrix} c_1 e^{i} & s_1 e^{-i} & 0 & 0 \\ -s_1 e^{i} & c_1^{-i} & 0 & 0 \\ 0 & 0 & c_1 e^{i} & s_1 e^{-i} \\ 0 & 0 & -s_1 e^{i} & c_1^{-i} \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \\ = \begin{bmatrix}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

Starprezultāta reizinājums ar otro kāpņveida matricu veikts (2.14) izteiksmē. Šādā veidā iegūtā spektra secību uzskatīsim par pareizu.

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

Konveijera arhitektūras dekompozīcijas otrā kaskāde parādīta 2.9a. attēlā. Otrā kaskāde sastāv no diviem kompleksajiem pagriezējiem. Tieši tāpat kā pirmajā kaskādē (skat. 2.8. att.), no divām virknes nolasēm ieejā tiek iegūtas divas paralēlas nolases izejā. Abu otrās kaskādes pirmā kompleksā pagriezēja ieejas nolašu (y_{11} un y_{21}) iegūšanai nepieciešamas visas 4 ieejas signāla nolases (x_1, x_2, x_3, x_4) (2.13). Tieši tāpat arī otrajam pagriezējam.

Var uzskatīt, ka savienojot kaskādē kompleksos pagriezējus, notiek starprezultātu nolašu permutācija (līdzīgi kā (1.17), (1.18)). Grafisks permutāciju matricu attēlojums $N = 4$ gadījumam parādīts 2.9b. attēlā. Otrajā kaskādē četras nolases pa pāriem tiek pagrieztas ar diviem neatkarīgiem pagriezējiem, tāpēc, ņemot vērā iepriekš minēto, otrajai kaskādei tiek iegūta sekojoša sakarība:



(a) Otrā kaskāde

(b) Permutāciju matricu grafisks attēlojums ($N = 4$)

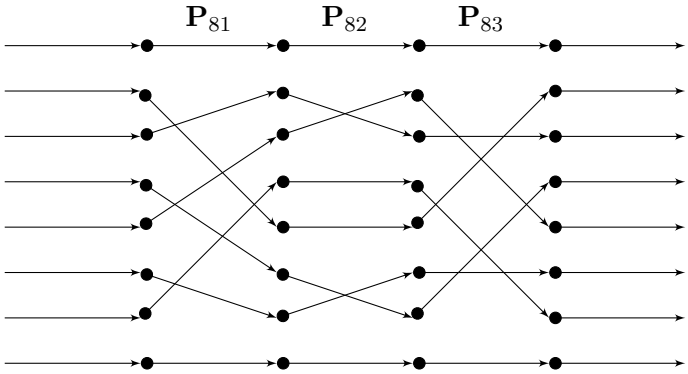
2.9. att. Četru nolašu dekompozīcija

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

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

Izmantojot izteiksmes (2.15) un (2.13), 2.5. attēlā redzamās struktūras rezultātus, varam iegūt secībā, kas ir ērta sistēmas darbības pārbaudei (nav jāveic permutācijas).

Palielinot kaskāžu skaitu n, palielināsies apstrādājamo nolašu skaits un mainīsies permutāciju matricu struktūra. 4−nolašu gadījumā abas permutāciju matricas ir vienādas, bet visiem pārējiem gadījumiem (ja $N > 4$), visas permutāciju matricas ir dažādas. Gra-



2.10. att. Permutāciju matricu grafisks attēlojums $N = 8$ gadījumam

fiski to struktūra $N = 8$ gadījumam parādīta 2.10. attēlā.

Izmantojot permutāciju matricas, iespējams iegūt ar konveijera arhitektūru realizēta pārveidojuma starprezultātus pēc katras kaskādes, par pamatu ņemot retinātās rotāciju matricas ar blokveida struktūru. Matemātiska starprezultātu iegūšana būtiski atvieglo parametriskā pārveidojuma izveidi. Blokveida rotācijas matricu iegūšanai var izmantot Kronekera reizinājumu:

$$\mathbf{B}_b(\Phi_1) = \mathbf{I}_4 \otimes \mathbf{T}_U(\Phi_1, ID) = \begin{bmatrix} \mathbf{T}_U(\Phi_1, ID) & \mathbf{0}_2 & \mathbf{0}_2 & \mathbf{0}_2 \\ \mathbf{0}_2 & \mathbf{T}_U(\Phi_1, ID) & \mathbf{0}_2 & \mathbf{0}_2 \\ \mathbf{0}_2 & \mathbf{0}_2 & \mathbf{T}_U(\Phi_1, ID) & \mathbf{0}_2 \\ \mathbf{0}_2 & \mathbf{0}_2 & \mathbf{0}_2 & \mathbf{T}_U(\Phi_1, ID) \end{bmatrix} \Bigg|_{ID=424} =$$

$$= \begin{bmatrix} c_1 e^{i} & s_1 e^{-i} & 0 & 0 & 0 & 0 & 0 & 0 \\ -s_1 e^{i} & c_1 e^{-i} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & c_1 e^{i} & s_1 e^{-i} & 0 & 0 & 0 & 0 \\ 0 & 0 & -s_1 e^{i} & c_1 e^{-i} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & c_1 e^{i} & s_1 e^{-i} & 0 & 0 \\ 0 & 0 & 0 & 0 & -s_1 e^{i} & c_1 e^{-i} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & c_1 e^{i} & s_1 e^{-i} \\ 0 & 0 & 0 & 0 & 0 & 0 & -s_1 e^{i} & c_1 e^{-i} \end{bmatrix} \tag{2.16}$$

Izmantojot permutāciju matricu P_{81}, var iegūt kāpņveida rotācijas matricu no retinātās blokveida matricas. Arī $N = 8$ gadījumam, ar konveijera arhitektūrā izveidotu signāla dekompozīciju iegūtā spektra koeficientu $\mathbf{Y}_b$ secība atšķirsies no tās, kura iegūta izmantojot 3 kāpņveida faktorizētās matricas. $\mathbf{Y}_b$ iegūst izmantojot $\mathbf{P}_{81}$ un $\mathbf{P}_{82}$:

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

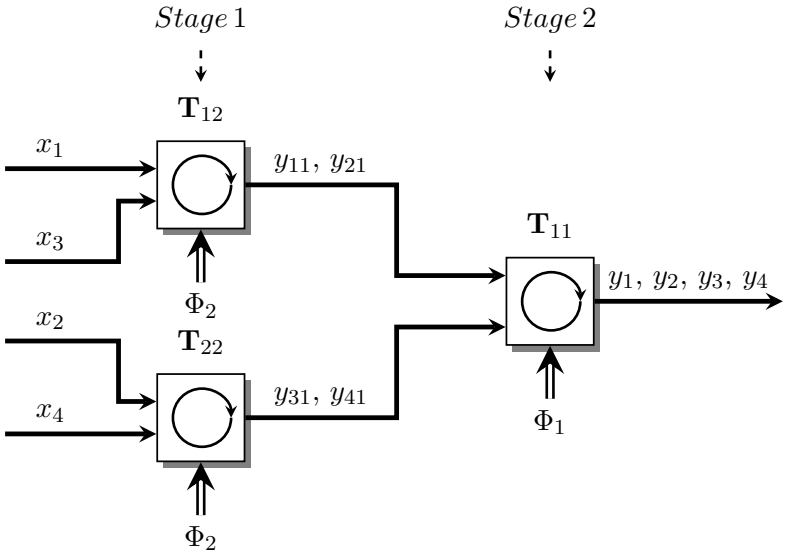
Vienkāršākam pierakstam, šajā nodaļā tika apskatīti tikai vienas klases (**CCRAIMOT**) ortogonālie pārveidojumi. Vispārīgā gadījumā (**CRABOT** klase), faktorizēto blokveida rotācijas matricu ($N = 4$ gadījumam) ar vispārināto Kronekera reizinājumu [3] var izteikt sekojoši:

$$\mathbf{B}_b(\mathbf{\Phi}_k) = \{\mathbf{I}_2\} \otimes \{\mathbf{T}_{U,1k}(\Phi_{1k}, ID_{1k}), \mathbf{T}_{U,2k}(\Phi_{2k}, ID_{2k})\} =$$
$$= \begin{bmatrix} \mathbf{T}_{U,1k}(\Phi_{1k}, ID_{1k}) & \mathbf{0}_2 \\ \mathbf{0}_2 & \mathbf{T}_{U,2k}(\Phi_{2k}, ID_{2k}) \end{bmatrix} \tag{2.18}$$

Jāņem vērā, ka pilnam ortogonālajam pārveidojumam nepieciešamas $n = log_2(N)$ retinātās rotācijas matricas.

2.3.2. Signāla rekonstrukcija

Vienkāršota signāla rekonstrukcijas ar *RE* blokshēma parādīta 2.11. attēlā. Ja dekompozīcijas gadījumā no virknes nolasēm tiek iegūtas paralēlas nolases (kokveida struktūra ar paplašinājumu), tad rekonstrukcijas gadījumā, no paralēlām nolasēm iegūstam virknes nolases (kokveida struktūra ar sašaurinājumu). Pie kam, katra *RE* virknes nolašu diskretizācijas frekvence ir divas reizes augstāka par paralēlo nolašu diskretizācijas frekvenci (neatkarīgi no tā, vai virknes nolases ir *RE* ieejā vai izejā).



2.11. att. Vienkāršota **CCRAIMOT** rekonstrukcijas blokshēma, $N = 4$

Rezumējums

Iegūtie rezultāti

- Izanalizēta daudzpakāpju parametriskas kokveida signālu dekompozīcijas-rekonstrukcijas sistēmas [P10], [P11] uzbūve un darbība.

Secinājumi

- Katrai unitārās elementārās rotācijas matricas struktūrai $\mathbf{T}_U(\phi, \psi, \gamma, ID)$ atbilst savs *RE*.
- *RE* veic divdimensionāla signālu vektora rotāciju, tāpēc katram *RE* nolases ieejā jāpadod paralēli, sadalītas pa $N = 2$ nolases gariem vektoriem.
- Sistēmas pārskaņošanu veic attiecīgajā laika momentā mainot *RE* parametrus.
- Iespējami vismaz divi arhitektūras varianti, kā realizēt *RE* – klasiskais algoritms ar reizināšanām un summēšanām, un *CORDIC* algoritms (skat. 3.3. nodaļu).

- Dekompozīcijas kokveida sistēmā, pēc katras kaskādes notiek nolašu ātruma pazemināšana (decimācija) divas reizes ($2x$ tiek pazemināta takts frekvence). Tas ļauj pazemināt nolašu apstrādes ātrumu.

- Rekonstrukcijas kokveida sistēmā, pēc katras kaskādes notiek nolašu ātruma palielināšana divas reizes ($2x$ tiek palielināta takts frekvence). Tas nozīmē, ka pēdējā kaskādē nolases ir jāapstrādā ar tādu pašu ātrumu kā dekompozīcijas sistēmas pirmajā kaskādē.

2.4. Signāla dekompozīcija, izmantojot kompleksus *FIR* filtrus

Iepriekšējā nodaļā apskatīto signālu dekompozīciju ar *RE*, varam realizēt arī ar kompleksiem *FIR* filtriem. Tāpēc, sākumā apskatīsim no rotācijas matricas (1.3) iegūto komplekso *FIR* filtru parametrus. Pēc tam, izveidosim dekompozīcijas-rekonstrukcijas sistēmu, un novērtēsim tās priekšrocības un trūkumus, salīdzinot to ar 2.3. nodaļā aprakstīto.

2.4.1. Komplekso filtru pārvades funkcijas

Ortogonālos filtrus var izmantot, lai ieejas signālu sadalītu komponentēs. Veivletu apstrādē, katrs ortogonālo filtru pāris, signālu sadala aproksimācijas (zemo frekvenču) un detalizācijas (augsto frekvenču) daļās. Ortogonālos filtrus izmanto ne tikai Veivletu apstrādē, bet arī, piemēram, Hilberta transformācijas veikšanai [20], kur vienotā ortogonālu filtru sistēmā apvienoti kompleksie un reālie *FIR* filtri.

Kompleksi ortogonāli filtri un to īpašības apskatīti [8]. Laikā mainīgu dažādu struktūru reālu filtru bankas apskatītas [21], [22] un [23] un daudzās citās publikācijās. Ciparu filtru aprakstošā pārvades funkcija [24] tiek izteikta kā:

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

Zinot pārvades funkciju, varam iegūt filtra frekvenču pārvades raksturlīkni. Tā kā mūsu gadījumā tiks izmantoti FIR (galīgas impulsu reakcijas filtri), tad frekvenču pārvades raksturlīkni iegūst sekojošā veidā:

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

Frekvenču pārvades raksturlīkne ir kompleksa. Tās moduli (visbiežāk izsaka decibelos) sauc par filtra amplitūdas frekvenču raksturlīkni. Filtra fāzes frekvenču raksturlīkni iegūst sekojošā veidā:

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

Komplekso rotācijas matricu (1.6) interpretējot kā divus filtrus (ar tiem sekojošu decimāciju), kur pirmā matricas (1.6) rinda atbilst 1.1. attēlā redzamās blokshēmas **Lo** aproksimācijas filtram un otrā rinda **Hi** detalizācijas filtram, izmantojot (2.20) un (2.21) varam iegūt abu filtru amplitūdas un fāzes frekvenču pārvades raksturlīknes. Piemēram, **Lo** filtra frekvenču pārvades rakturlīkni iegūst pēc sekojošas sakarības:

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

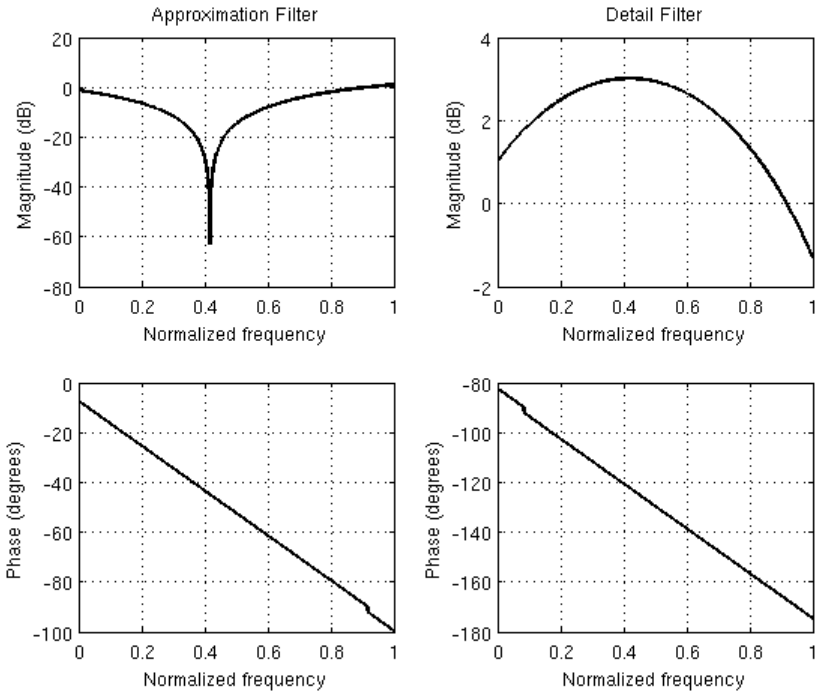
kur $\omega \in [0,\pi]$, (ϕ, ψ, γ) – filtra parametri

Brīvi izvēloties filtra parametrus $\Phi_k = (\phi_k, \psi_k, \gamma_k) = \left(\frac{\pi}{4}, \frac{\pi}{3}, \frac{\pi}{4}\right)$ iegūstam attiecīgo aproksimācijas un detalizācijas filtru amplitūdas un fāzes frekvenču raksturlīknes[1] (skat. 2.12. att.).

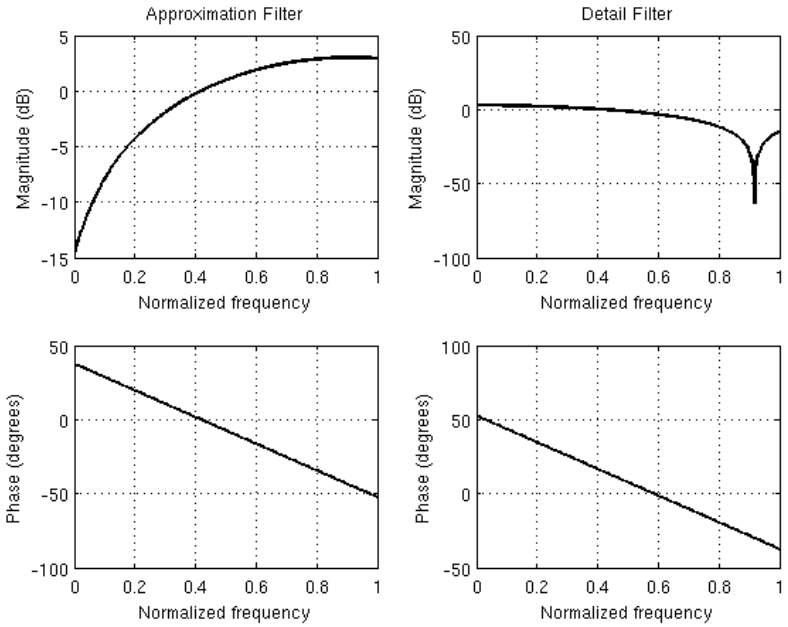
Ņemot transponēto matricu (1.12) iegūstam rekonstrukcijas filtrus (2.13. attēls).

Apvienojot vienotā sistēmā no tiešās rotācijas iegūto filtru pāri un filtru pāri, kas iegūts no transponētās rotācijas matricas, izveidojam dekompozīcijas – rekonstrukcijas sistēmu (2.14. att.), kas ļauj pilnībā rekonstruēt signālu. Lai varētu veikt bezzudumu

[1] Frekvenču vērtības (x−ass) ir normētas pret $\frac{Fs}{2}$



2.12. att. Aproksimācijas un detalizācijas **De**kompozīcijas filtru amplitūdas un fāzes frekvenču raksturlīknes, $\phi = \frac{\pi}{4}$, $\psi = \frac{\pi}{3}$, $\gamma = \frac{\pi}{4}$



2.13. att. Aproksimācijas un detalizācijas **Re**konstrukcijas filtru apmlitūdas un fāzes frekvenču raksturlīknes, $\phi = \frac{\pi}{4}$, $\psi = \frac{\pi}{3}$, $\gamma = \frac{\pi}{4}$

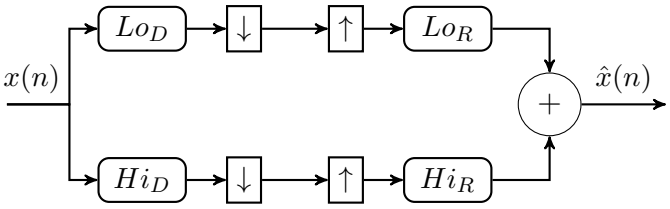
rekonstrukciju, jāizpildās sekojošam nosacījumam [22]:

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

Mainot parametrus, iespējams iegūt praktiski neierobežotu skaitu dažādu, gan reālu, gan kompleksu, pirmās kārtas ortogonālu *FIR* filtru pāru (skat. 2.15. att.). Kompleksu filtru gadījumā (skat. 2.15b. att.), var izdalīt kopējas tendences amplitūdas frekvenču pārvades raksturlīkņu atkarībai[1] no leņķiem/parametriem:

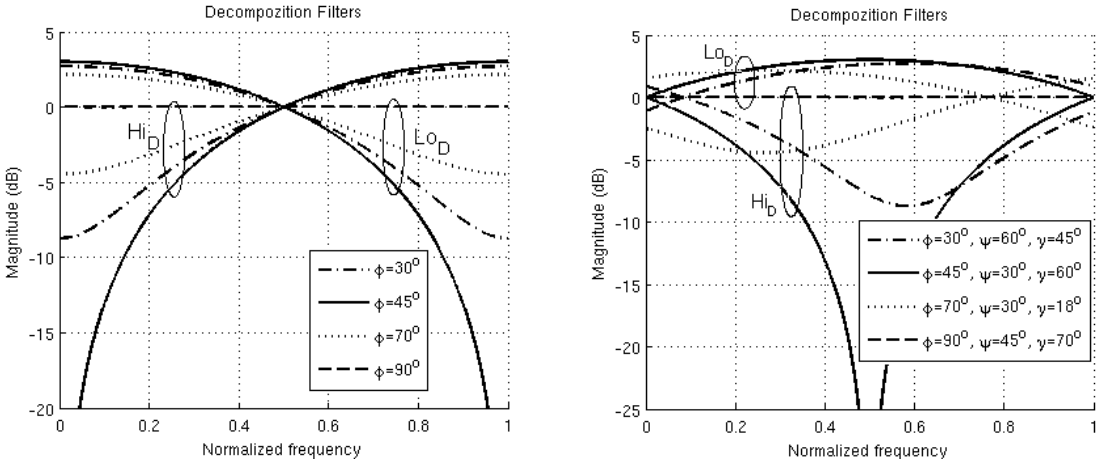
- Komplekso filtru apmlitūdas frekvenču raksturlīknes ir daudz komplicētāka par reālu filtru amplitūdas frekvennču raksturlīknēm,
- Komplekso aproksimācijas/detalizācijas filtru (**Lo**/**Hi**) amplitūdas frekvenču raksturlīknēm var būt gan joslu/režekcijas, gan režekcijas/joslu filtru forma (atkarībā no elementārās rotācijas matricas *ID*),

[1]Nedrīkst aizmirst, ka pēc filtrēšanas seko decimācija



2.14. att. De–Re filtri

- leņķis ϕ nosaka joslu/režekcijas filtru joslu platumu (precīzas sakarības pagaidām nav iegūtas),
- leņķi ψ un γ nosaka joslu/režekcijas filtru caurlaides/režekcijas joslas frekvenci (precīzas sakarības pagaidām nav iegūtas).



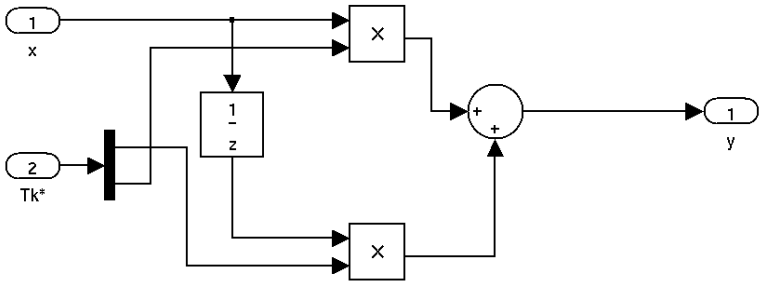
(a) Reāli filtri ($\mathbf{T}_U(\phi, 0, 0, 812)$)

(b) Kompleksi filtri ($\mathbf{T}_U(\phi, \psi, \gamma, 812)$)

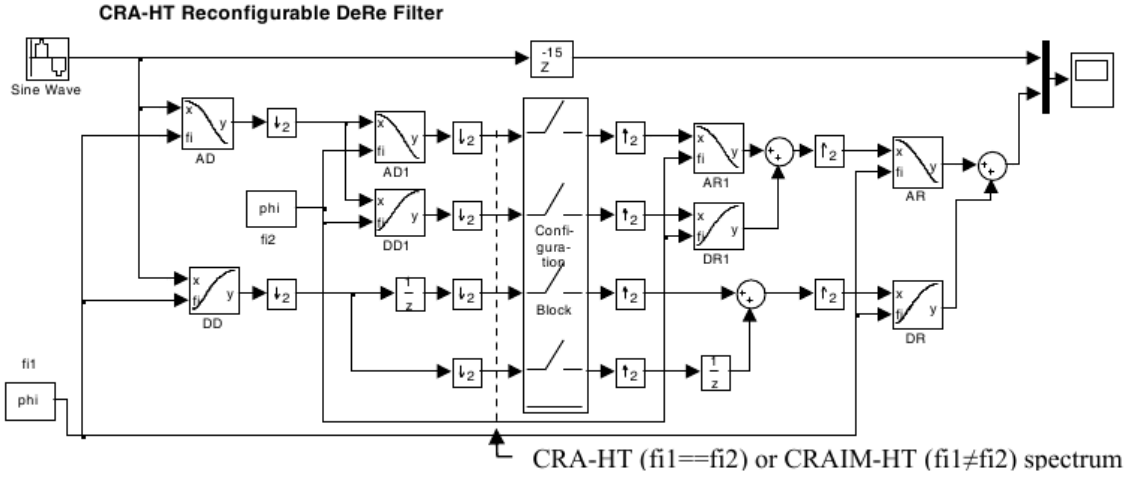
2.15. att. Dekompozīcijas aproksimācijas un detalizācijas filtru amplitūdas pārvades raksturlīknes (daļēji ((a)) atrodamas [P3], [P6] un [P8])

2.4.2. Kompleksu ortogonālu filtru kaskāde

Izveidosim kompleksu *FIR* filtru (skat. 2.16. att.). Izveidotā filtra vienkāršots variants (reāli koeficienti, kas tiek aprēķināti filtra iekšienē) 2.17. attēlā parādītajā Hārveidīgajā dekompozīcijas-rekonstrukcijas sistēmā izmantots gan kā aproksimācijas dekompozīcijas (AD) un detalizācijas dekompozīcijas (DD), gan arī kā aproksimācijas rekonstrukcijas (AR) un detalizācijas rekonstrukcijas (DR) filtrs. Kompleksu *FIR* filtru realizācijai tiek izmantota tiešā veida (*Direct-form*) ciparu filtru struktūra [26]. Jāņem vērā, ka mēs veidojam



2.16. att. Kompleksā FIR filtra *Simulink* vienkāršots modelis

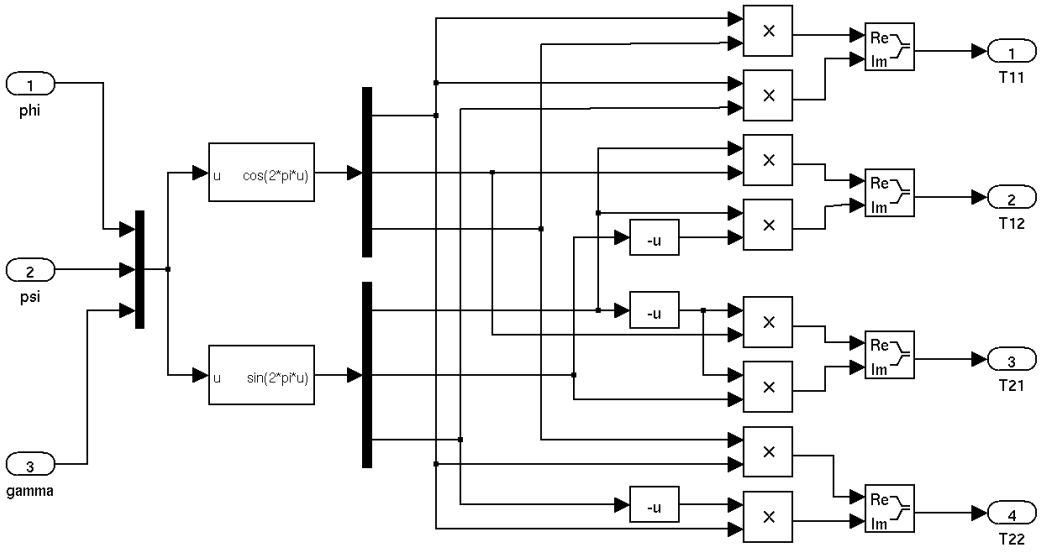


2.17. att. **RA-HT** *DeRe* filtru *Simulink* modelis [P3]

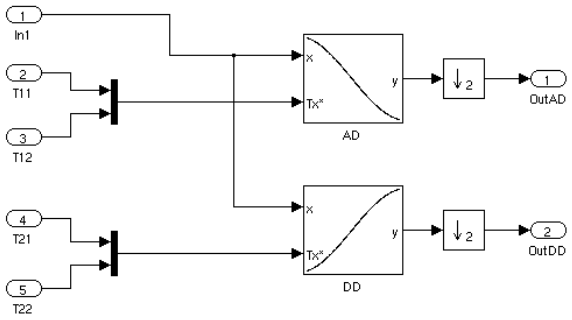
sistēmu, kurā ieejas signāls un visi filtru koeficienti ir kompleksi lielumi, kuru savstarpējā reizināšana ir pietiekoši sarežģīta operācija [27].

Tomēr lielākais šādi izveidotas sistēmas trūkums ir sarežģītā filtru pārskaņošana, jo filtru koeficienti tiek sarēķināti no trim reāliem parametriem – leņķiem ϕ, ψ un γ (2.17. attēlā parādītajā sistēmā, visu filtru koeficientu aprēķinam izmanto $\mathbf{T}_U(\phi, \psi = 0, \gamma = 0,424)$). Atkarībā no kompleksās rotācijas matricas struktūras (1.3), varam iegūt dažādus filtru koeficientus. Filtru koeficientu iegūšana vienai struktūrai parādīta 2.18. attēlā.

Par šādas sistēmas priekšrocību var uzskatīt to, ka tā nemainās atkarībā no komplekso rotācijas matricu struktūrām - mainās tikai filtru koeficienti. Tomēr, kā redzams



2.18. att. Kompleksās matricas $\mathbf{T}_U(\phi, \psi, \gamma, 424)$ (filtru koeficientu) iegūšanas *Simulink* modelis

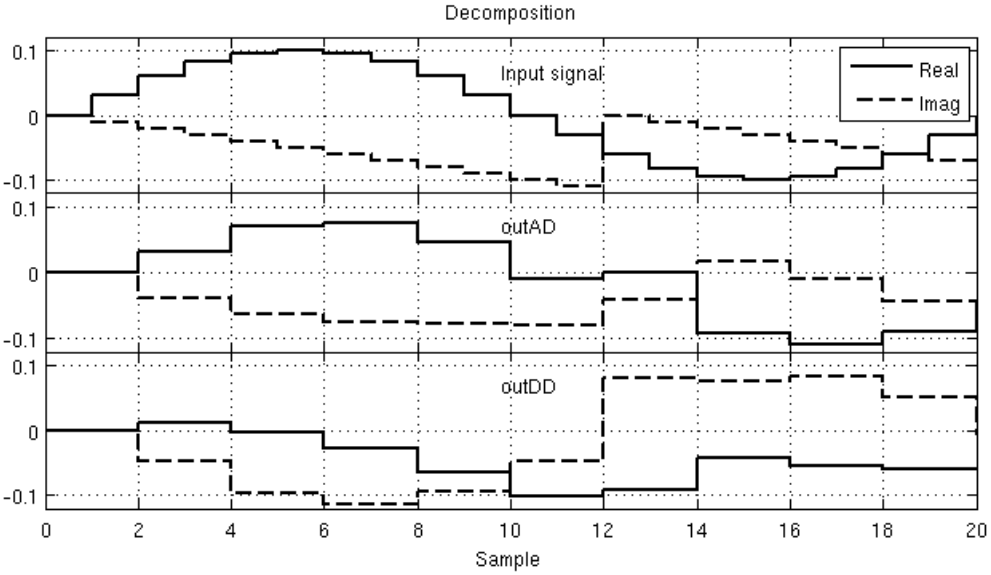


2.19. att. No diviem *FIR* filtriem veidotas dekompozīcijas kaskādes *Simulink* modelis

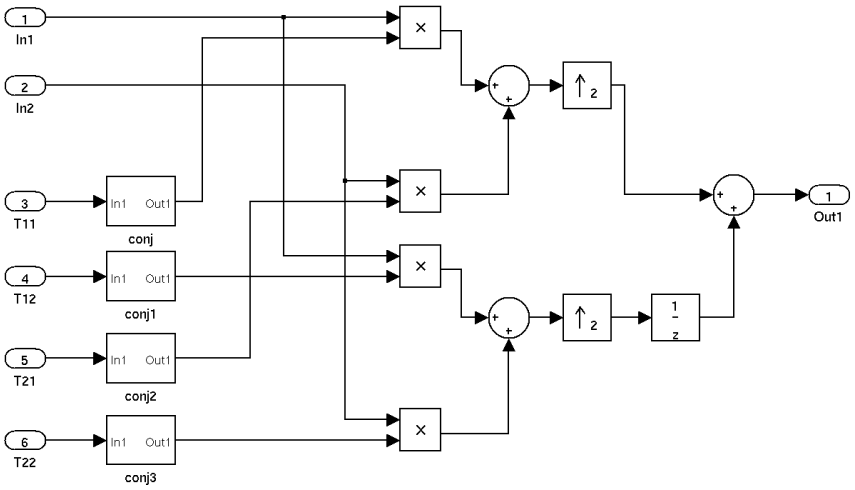
2.18. attēlā, ortogonālu filtru koeficientu sarēķināšana ir relatīvi sarežģīts uzdevums, lai pārskaņojamu, šādā veidā realizētu sistēmu, uzskatītu par optimālu.

Izmantojot filtrus, dekompozīcijas-rekonstrukcijas sistēmas veidošanā, nedrīkst aizmirst par decimāciju (*downsampling*). Sistēmu veidojot no diviem *FIR* filtriem, diskretizācijas frekvence jāsamazina divas reizes. 2.19. attēlā parādīts pirmās kaskādes *Simulink* modelis signālu dekompozīcijas veikšanai. *Simulink* laika diagrammas, ja ieejas signāla reālā daļa ir sinusoidāla, bet imaginārā - zāģveida, parādītas 2.20. attēlā.

Signāla rekonstrukcijai jāizmanto kompleksās $\mathbf{T}_U$ matricas transponētā matrica - jāņem koeficientu kompleksi saistītie lielumi un jāmaina rindas ar kolonnām vietām, kā arī jāveic diskretizācijas frekvences palielināšana divas reizes (*upsampling*). Signāla rekonstrukcijas *Simulink* modelis parādīts 2.21. attēlā, kur *In1* ir dekompozīcijas modeļa *outAD* signāls, savukārt *In2*–dekompozīcijas modeļa *outDD* signāls.



2.20. att. Ieejas signāla dekompozīcijas *Simulink* laika diagrammas, $[\phi = \frac{\pi}{4}, \psi = \frac{\pi}{3}, \gamma = \frac{\pi}{6}]$



2.21. att. Signāla rekonstrukcijas *Simulink* modelis

Rezumējums

Rezultāti

- Ir provizoriski izanalizēta uz *FIR* filtriem balstītas alternatīvas dekompozīcijas-rekonstrukcijas sistēmas piemērotība realizācijai.

Secinājumi

- Filtru struktūra nemainās, mainot sistēmas parametrus.
- sistēma darbojas ar virknes ieejas datiem (ieejas nolases nav jāsadala pa N-nolašu blokiem).
- Visi pirmās kārtas kompleksie filtri vienmēr sastāv no 16 reizinātājiem un 12 summatoriem, kas neļauj veikt vienkāršojumus.
- Kompleksu filtru gadījumā, katrs no abu filtru (**Lo** un **Hi**) koeficientiem ir atkarīgs no visiem trijiem leņķiem/parametriem, kas sistēmas pārskaņošanu padara sarežģītu un resursietilpīgu.
- Reālu filtru koeficienti atkarīgi tikai no viena leņķa ϕ, kas sistēmu padara relatīvi viegli pārskaņojamu.
- Turpmākos 1-D signālu pārveidojumu pētījumos tiek izmantoti kompleksi lielumi, tāpēc, ņemot vērā iepriekšējos secinājumus, signālu dekompozīcija, izmantojot kompleksus filtrus, pagaidām atzīta par neefektīvu.
- Varam uzskatīt, ka, veicot dekompozīciju/rekonstrukciju arī ar *RE*, ar ieejas signālu tiek veiktas pieminētās manipulācijas – filtrācija ar **Lo** un **Hi** filtriem un decimācija[1].

[1]Rekonstrukcijas gadījumā, atbilstoši, diskretizācijas frekvences palielināšana

3. Algoritmu realizācija FPGA

1. un 2. nodaļās apskatīto algoritmu realizācija FPGA mikroshēmās ir nozīmīga šī darba daļa. Šajā nodaļā tiks apskatīti dažādi realizācijas veidi un ar tiem saistītās problēmas. Izvērtējot visu 2. nodaļā aprakstīto arhitektūru variantu priekšrocības un trūkumus, turpmāk ir nolemts darboties ar no *RE* veidotām dekompozīcijas-rekonstrukcijas sistēmām.

3.1. Peldošā un fiksētā punkta aritmētika

Lai arī peldošā punkta aritmētika (turpmāk - peldošais punkts) arvien vairāk tiek realizēta FPGA mikroshēmās [35]-[38], šajā darbā aplūkotajos algoritmos tiek izmantota fiksētā punkta aritmētika (turpmāk - fiksētais punkts), kas arī turpmāk pastāvēs paralēli peldošā punkta aritmētikai [39]. Peldošā punkta aritmētika neatrisina visas ar precizitāti saistītās problēmas [40], bet padara sarežģītāku, piemēram, *VHDL* kodu simulāciju.

Algoritmu realizēšana, izmantojot fiksētā punkta aritmētiku, vienmēr saistās ar fiksētā punkta kļūdām [41], kuru eksperimentālais novērtējums *RE*-am sniegts 3.5. nodaļā.

3.2. DSP elementu realizācija FPGA

3.2.1. Reizinātāja ar dažādu izejas vārda garumu realizācija *FPGA*

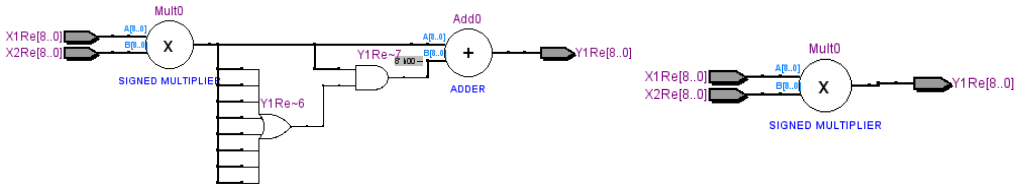
Realizējot reizinātāju *FPGA*, izmantojot fiksētā punkta aritmētiku (praktiski visiem jaunāko paaudžu Altera's *FPGA* ir iebūvēti fiksētā punkta reizinātāji [42]), tā izejā signāliem vienmēr būs divreiz lielāks vārda garums (WL) nekā ieejā. No vairākiem reizinātājiem sastāvošos algoritmos tas radītu virkni problēmu, jo notiktu pārmērīga signālu WL palielināšanās, tāpēc parasti tiek veikta mērogmaiņa (*scaling*), kas ietver sevī bitu izdalīšanu. Q1.x aritmētikā [43], samazinot bitu skaitu līdz w bitiem (izdalot vecākos w bitus), samazinās rezultāta precizitāte, kuru nosaka kvantēšanas solis [43]:

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

kur x – $Q1.x$ *FPA* daļskaitļa bitu skaits ($x = w - 1$).

Bitu izdalīšanu var veikt, izmantojot četrus dažādus noapaļošanas veidus:

- floor – rezultāts vienmēr tiks noapaļots virzienā uz $-\infty$ pusi līdz ekvivalentajam veselajam,
- fix – rezultāts tiks noapaļots virzienā uz nulli līdz ekvivalentajam veselajam,
- ceil – rezultāts vienmēr tiks noapaļots virzienā uz $+\infty$ pusi līdz ekvivalentajam veselajam,
- round – rezultāts tiks noapaļots uz tuvāko ekvivalento veselo.



(a) *fix* noapaļošana

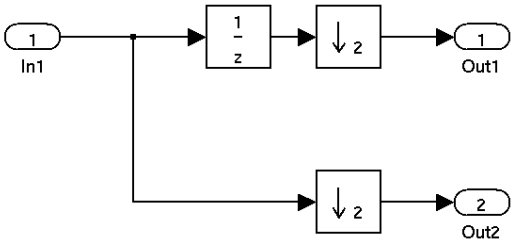
(b) *floor* noapaļošana

3.1. att. Ar *HDL Coder* ģenerētu reizinātāju ar bitu izdalīšanu *RTL* attēlojums

Katram no šiem noapaļošanas veidiem atšķiras realizācija un līdz ar to arī papildus izmantojamo loģisko elementu skaits. Izmantojot *Simulink HDL coder* un *MatLab* ***fi()*** *object* viegli var izvēlēties vēlamo noapaļošanas veidu. *Fix* un *floor* noapaļošanas "aparatūriskais" rezultāts, ja bitu skaits tiek samazināts no 16 uz 8 parādīts 3.1. attēlā. *Floor* ir vienīgais noapaļošanas veids, kuram nav vajadzīgi papildus loģiskie elementi (skat. 7. tabulu).

3.2.2. Virknes–paralēlais un paralēlais–virknes datu pārveidotājs

Dažādo pagriezēju struktūru sintēzes automatizācijai, aritmētiskās darbības tiek pierakstītas izteiksmju (pārveidotu matricu) veidā, t.i., 2 nolašu dati ieejā ir paralēli. Tāpēc



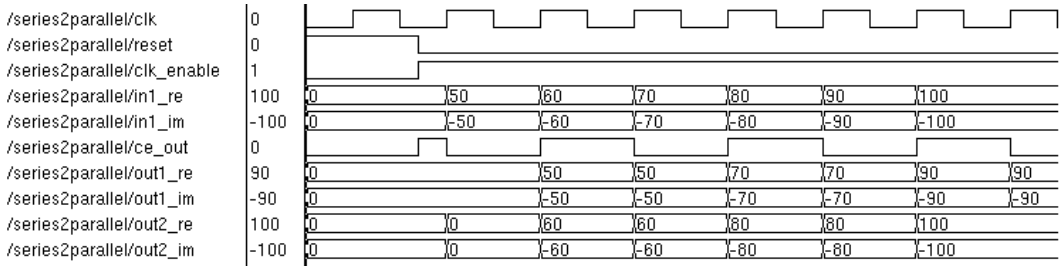
3.2. att. Virknes–paralēlais datu pārveidotājs (*Simulink* blokshēma)

reālās iekārtās, pirms katra pagriezēja jābūt pārveidotājam, kas virknē ienākošos datus pārveido par paralēliem datiem. Šādu pārveidotāju, protams, var uzrakstīt "pa tiešo" *VHDL* valodā, bet ērtāk un uzskatāmāk ir izmantot *Simulink* un *HDL coder*, lai no 3.2. attēlā redzamās blokshēmas iegūtu attiecīgo *VHDL* kodu.

Uzģenerētā *VHDL* koda darbības laika diagramma, kas iegūta ar *ModelSim*, redzama 3.3. attēlā. *ModelSim* ir plaši izmantota [28]-[34] *HDL* kodu simulācijas programma. Sīkāk

6. tabula. Loģisko elementu skaits

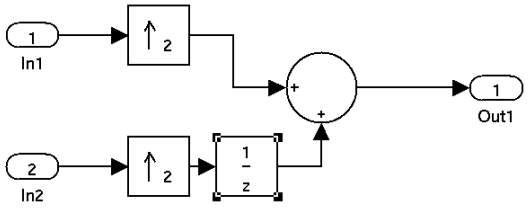
	S2P	P2S
LUT's	50	49
ALUT's	34	18



3.3. att. Virknes–paralēlā datu pārveidotāja laika diagramma

VHDL kodu iegūšana, izmantojot *Simulink HDL Coder*, apskatīta 3.6. nodaļā. *HDL* koderis, papildus kompleksajai ieejai un divām izejām, automātiski pievieno standarta ciparu iekārtu izvadus – takts impulsu *clk*, darbību atļaujošo *clk_enable* un nomešanas izvadu *reset*.

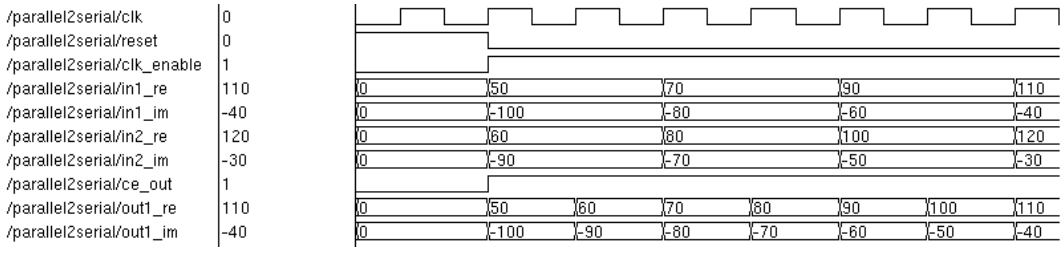
Virknes datus no paralēliem iegūst tieši tādā pašā veidā – izveidojot pretējas darbības *Simulink* blokshēmu (skat. 3.4. att.). Paralēlā–virknes datu pārveidotāja *ModelSim* laika



3.4. att. Paralēlais–virknes datu pārveidotājs (*Simulink* blokshēma)

diagramma attēlota 3.5. attēlā.

Programmējamie elementi uz dažādām *FPGA* saimēm atšķiras, tāpēc dažāds ir arī loģisko elementu skaits, kāds nepieciešams vienam un tam pašam algoritmam. Uz *Altera*'s ražotajiem *FPGA* iespējami divi programmējamās loģikas varianti: ***LUT**-Look Up Table* un ***ALUT**-Adaptive Look–Up Table* [44]. ALUT priekšrocība ir tā, ka tos var efektīvāk apvienot grupās. 6. tabulā parādīts LUT un $ALUT$ skaits, kāds nepieciešams, lai realizētu virknes−paralēlo (*S2P*) un paralēlo−virknes (*P2S*) pārveidotāju.



3.5. att. Paralēlā–Virknes datu pārveidotāja laika diagramma

Rezumējums

Šajā nodaļā ir apskatīti jautājumi, kas, autoraprāt, atvieglo kopsavilkuma materiāla izpratni un nav akcentējami kā rezultāti, bet sasaucas ar apakšnodaļu 3.3 un 3.5 saturu.

Secinājumi

- Lai pārmērīgi nepalielinātos signāla nolašu WL, pēc reizināšanām jāveic bitu izdalīšana,
- Noapaļošanas veids pie bitu izdalīšanas būtiski ietekmē izmantojamo LE skaitu (sk. 3.3. apakšnodaļu) un elementārā spektra aprēķina kļūdu (sk. 3.5. apakšnodaļu).
- *Altera*s *FPGA* LE sastāvā esošo loģisko kombināciju shēmu veids (LUT vai $ALUT$) ietekmē izmantojamo LE skaitu – ja tiek izmantoti LE ar $ALUT$, tad nepieciešams mazāks LE skaits (vidēji par $\approx 25\%$).

3.3. Vispārinātā Jakobi pagriezēja realizācija

Šajā nodaļā apskatīti divi iespējamie veidi (klasiskais un ar *CORDIC* algoritmu) kā *FPGA* mikroshēmās iespējams realizēt komplekso Jakobi pagriezēju, kurš tiek aprakstīts ar komplekso rotācijas matricu (1.3).

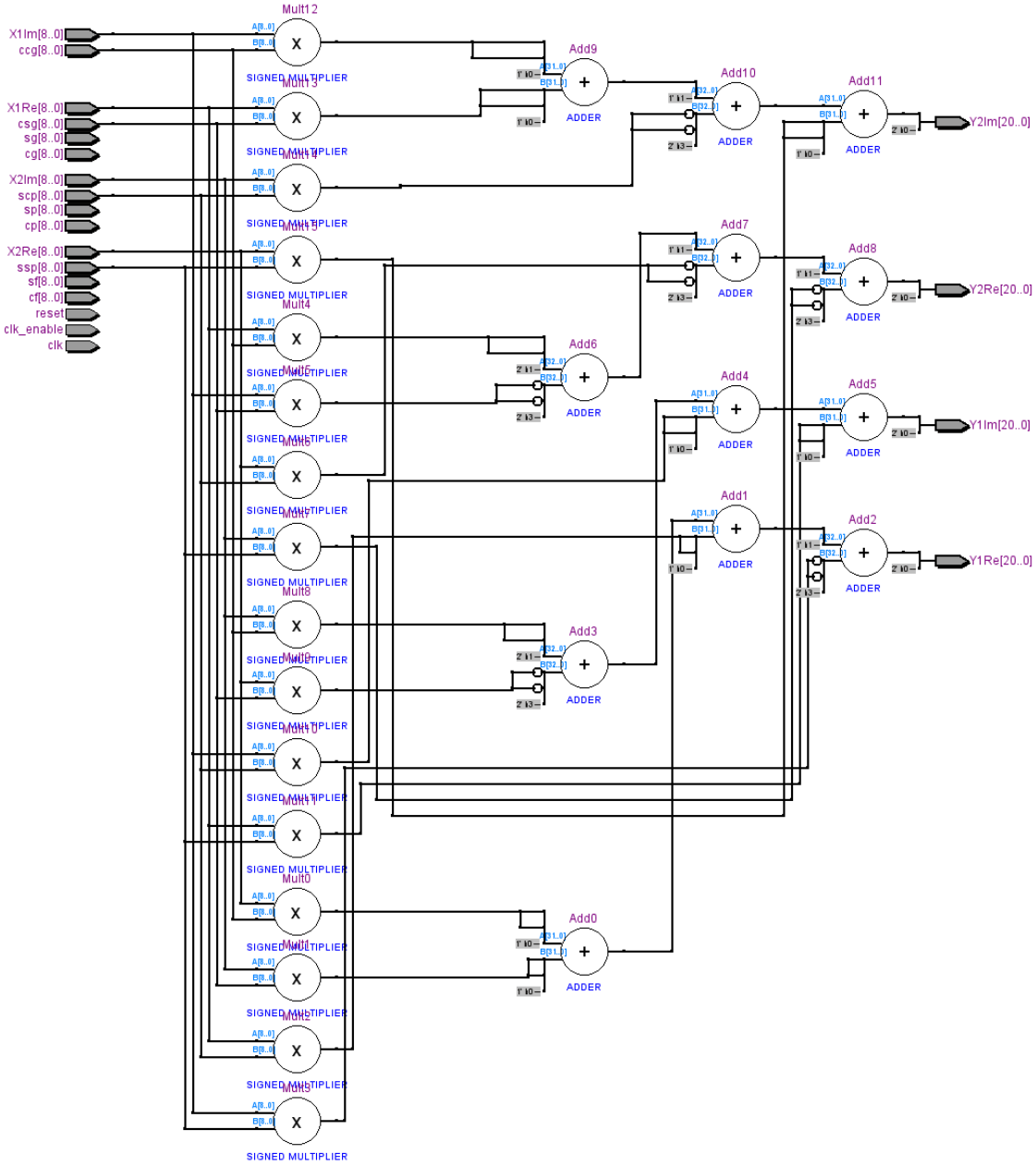
3.3.1. Klasiskā realizācija, izmantojot reizinātājus un summatorus [P9]-[P12]

Vienas pilnas kompleksās rotācijas realizācijai nepieciešami 16 reizinātāji un 12 summatori (3.6. att.), ja izmantojam atsevišķi sarēķinātus sinusu un kosinusu savstarpējos starpreizinājumus (3.3). Zinot, ka reālās sistēmās, kompleksa signāla reālā un imaginārā daļa tiek apstrādātas atsevišķi, kompleksa signāla reizinājumu ar kompleksu rotācijas matricu $\mathbf{T}_U(\phi,\ \psi,\ \gamma, 424)$ varam pārrakstīt, sadalot to reālajā un imaginārajā komponentēs:

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

kur

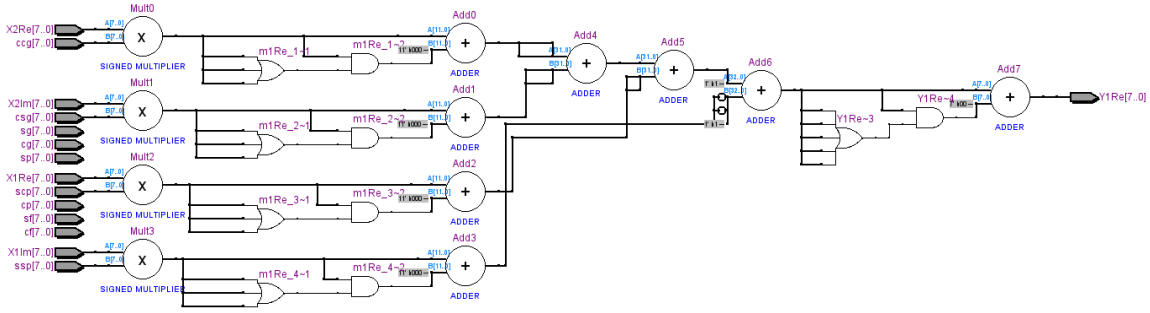
$$\begin{aligned} ccg &= \cos(\phi) \cdot \cos(\gamma), \quad csg = \cos(\phi) \cdot \sin(\gamma), \\ scp &= \sin(\phi) \cdot \cos(\psi), \quad ssp = \sin(\phi) \cdot \sin(\psi). \end{aligned} \tag{3.3}$$



3.6. att. Jakobi pagriezēja *RTL* attēlojums (neveicot bitu izdalīšanu)

3.7. attēlā, vienkāršības labad, parādīts tikai Y_{Re} komponentes iegūšanas *RTL* attēlojums ar *fix* noapaļošanu.

Algoritmus realizējot reālās iekārtās, svarīgs ir skaitļošanas laika novērtējums - laiks, kāds nepieciešams, lai signāls no ieejas nonāktu izejā. *FPGA* gadījumā, jānovērtē arī izmantojamo loģisko elementu skaits. 7. tabulā apkopoti signālu aizturu laiku un izmantojamo loģisko elementu skaits atkarībā no noapaļošanas veida un signālu nolašu *WL*. Ar w_{in} jāsaprot ieejas signāla *WL*, w_{mult} – *WL* starprezultātam pēc reizinātāja (neveicot bitu atmešanu, tas ir $w_{mult} = 2 \cdot w_{in}$) un w_{out} – izejas signāla *WL*. Jāņem vērā, ka noapaļošanas



3.7. att. Y_{Re} iegūšanas *RTL* attēlojums ar *fix* noapaļošanu ($w_{in} = 8$, $w_{mult} = 12$, $w_{out} = 8$)

veida un/vai signālu *WL* izmaiņas ietekmē arī *FPA* kļūdu (*FPA* kļūdas sīkāk apskatītas 3.5. nodaļā).

Tā kā parametriskie uz leņķiem bāzētie pārveidojumi sastāv no vairākiem *RE* (skat. 2.3. nodaļu), tad svarīgi ir nodrošināt minimālu *FPGA* resursu patēriņu katram *RE*. Šī

7. tabula. Sliktākā gadījuma *TPD* un *LE* skaits, *EP2C35F672C6* (tabula ņemta no [P12], papildināta)

	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

iemesla dēļ, par optimālāko noapaļošanas veidu, veicot bitu izdalīšanu, ir atzīta *floor* noapaļošana (skat. 7. tabulu), kas ir arī visātrākais no "aparatūriskajiem" noapaļošanas veidiem.

3.3.2. Realizācija ar *CORDIC*

Kompleksās rotācijas realizācija ar *CORDIC* algoritmu uz doto brīdi ir apskatīta tikai teorētiski [45], tāpēc nav iespējams veikt precīzu nepieciešamo loģisko elementu skaita salīdzinājumu ar klasisko (skat. 3.3.1. nodaļu) realizācijas veidu. Programmējamās loģikas noslogojumu principā var novērtēt no reālās *CORDIC* algoritma realizācijas [P4]-[P7], [P9], jo vienu komplekso rotāciju var veikt sadalot to četrās reālās rotācijās [45].

CORDIC algoritma pamatā [46] ir elementāro rotācijas izteiksmju pārveidošana vēlamā veidā. Pārveidojot elementārās rotācijas izteiksmes, iegūst

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

kur $K_i = \cos(\tan^{-1} 2^{-i}) = \frac{1}{\sqrt{1+2^{-2i}}}$,
$d_i = \pm 1$ (nosaka vektora pagriešanas virzienu)

Vektora pagriešanas leņķis atkarīgs no iterācijas kārtas numura:

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

CORDIC algoritma realizācijai *FPGA* tiek izmantoti divi paņēmieni:

1. Izvērstais (*unrolled*) – visu *CORDIC* iterāciju elementi (rotēšanas virziena noteikšana, bitu nobīde, summēšana) fiziski atrodas uz *FPGA*. Mainoties iterāciju skaitam (precizitātei), būtiski izmainās izmantoto *LE* skaits. Šādā veidā realizētam *CORDIC* algoritmam ir vienkāršāka tā apkalpošana, bet rezultāta iegūšanas laiks ir atkarīgs no iterāciju skaita un *FPGA* ātrdarbības.

2. Iteratīvais (*iterative*) – uz *FPGA* atrodas tikai vienas *CORDIC* iterācijas elementi, caur kuru uz katru takts impulsu, tiek sūtīti iepriekšējā taktī iegūtie rezultāti. Šādā veidā realizētam *CORDIC* algoritmam, rezultāta iegūšanai nepieciešamas vairākas taktis, bet katras iterācijas ilgums atkarīgs tikai no *FPGA* ātrdarbības klases. Iterāciju skaits praktiski neietekmē izmantoto *FPGA LE* skaitu.

8. tabula. *CORDIC* algoritma *FPGA* realizācijas parametri [*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

8. tabulā apkopoti ar *CORDIC* algoritma realizāciju *FPGA* mikroshēmās saistītie lielumi. Atkarībā no izvēlētās arhitektūras (izvērstā vai iteratīvā) un *CORDIC* iterāciju skaita, mainās izmantojamo *LE* skaits, kā arī maksimālā takts impulsu frekvence. Ja izvērstās arhitektūras gadījumā, pēc tabulā parādītā *Ts* tiks iegūts pagrieztais vektors, tad iteratīvās arhitektūras gadījumā – vienas iterācijas rezultāts. Pareizinot *Ts* ar attiecīgo iterāciju skaitu, aprēķinam laiku, pēc kura tiek iegūts iteratīvās arhitektūras *CORDIC* rezultāts. Tabulā parādīta arī maksimālā iespējamā *CORDIC* algoritma leņķu kļūda, kas izteikta procentos attiecībā pret *CORDIC* darbības diapazonu $[-90°, 90°]$.

Tikai *CORDIC* algoritma izmantošana uz rotācijas leņķiem bāzētu pārveidojumu veidošanā, no *FPGA* resursu izmantošanas viedokļa nav pats efektīvākais variants, jo *FPGA* paralēli *LE* satur arī *DSP* elementus [42], kuri var veikt salīdzinoši ātras ($\sim$ 20 ns uz *EP2C35F672C6*) *FPA* reizināšanas darbības. Pārveidojumus veidojot konveijera arhitektūrā, vislielāko resursu izmantošanas efektivitāti iespējams panākt pirmajās kaskādēs (skat. 2.3. nodaļu), izmantojot algoritmus ar *DSP* elementiem (to ātrdarbības dēļ), pārējās – *CORDIC* (nesatur *DSP* elementus, kas uz *FPGA* ir ierobežotā skaitā) un algoritmu ar *DSP* elementiem kombinācijas.

Rezumējums par [P5], [P6], [P9]-[P12]

Iegūtie rezultāti

- Ir izanalizēti divi kompleksās elementārās rotācijas veidi īstenošanai *FPGA*:
  - klasiskais – ar reizinātājiem un summatoriem,
  - ar *CORDIC*.

Secinājumi

- Rotācijas veikšanai izmantojot *CORDIC* algoritmu, netiek izmantoti *DSP* elementi,
- 3 parametru (ϕ, ψ un γ) kompleksās rotācijas veikšanai (ja visi parametri ir mainīgi), izmantojot klasisko algoritmu, nepieciešamas 20 reizināšanas un 12 summēšanas.
- Jebkuras arhitektūras *CORDIC* algoritms (ar piecām vai vairāk iterācijām – konkrēti, čipam *EP2C35F672C6*) ir lēnāks par vektora rotāciju, kas izmanto *DSP* reizinātājus. Šī īpašība ir raksturīga visiem iebūvētos reizinātājus saturošajiem *Alteras* čipiem.
- Apmierinošu rotācijas precizitāti panāk ar vismaz 9 *CORDIC* iterācijām, t.i., ar 9 iterācijām maksimālā leņķa kļūda ir 0.12%, bet, piemēram, ar 13 – 0.008%.
- *DSP* elementu skaits uz *FPGA* ir stipri ierobežots – izmantojot visus brīvos *DSP* elementus, reizinātāji tiks veidoti no *LE*-iem, kas būtiski ($\approx$ 110 *LE* uz vienu 9-bitu *DSP* elementu) palielinās to patēriņu.

- Kokveida algoritmu pirmajās kaskādēs, izmantojot *RE* klasisko realizāciju, panāk vislielāko ātrdarbību, bet *CORDIC* algoritma lietošana samazina izmantojamo *DSP* elementu skaitu. Vispārīga rekomendācija ir, ka ierīcēs ar maksimāli iespējamo ātrdarbību pirmajā(s) (rekonstrukcijas daļā pēdējā(s)) kaskādē(s) tiek rekomendēts klasiskais algoritms, bet pārējās kaskādēs *CORDIC*.

- Izmantojot klasisko algoritmu, noapaļošanas veida izvēle pie bitu izdalīšanas, ietekmē gan izmantojamo *LE* skaitu, gan algoritma ātrdarbību – *floor* noapaļošanai nav nepieciešami papildus *LE* un tas ir visātrdarbīgākais. Piemēram, salīdzinot *floor* ar *round* noapaļošanu, ja signāla nolašu *WL* ir ($w_{in} = 8$, $w_{mult} = 8$, $w_{out} = 8$), tad *floor* noapaļošanai nepieciešami 3 reizes mazāk *LE* un rezultāts tiek iegūts par 20 % īsākā laikā (*EP2C35F672C6*).

- Lai gan kompleksā rotācija ar *CORDIC* pagaidām *EGURIT* nav iebūvēta, ir zināms, ka komplekso rotāciju var veikt izmantojot 4 reālās rotācijas (t.i. reālos *CORDIC*-us) un tas nozīmē, ka darbā iegūtos rezultātus par reālā *CORDIC*-a resursu patēriņu principā var izmantot arī *CORDIC* balstītām kompleksajām rotācijām.

3.4. Vispārinātās kompleksās elementārās rotācijas matricas vienkāršojumi

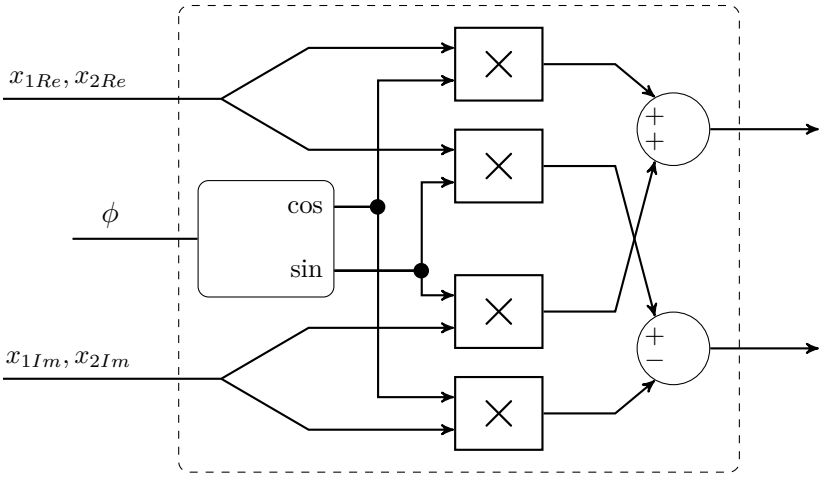
Pētījumi rāda, ka ne visos gadījumos nepieciešams realizēt "pilnu" 3 parametru komplekso rotācijas matricu (*EGURM*–šajā apakšnodaļā vienkārši matrica). Dažādiem pārveidojumiem pietiek ar matricas vienkāršotajiem variantiem, aizstājot kādu no parametriem ar konstantu lielumu. Tā, piemēram, ja $\psi = \frac{\pi}{2}$ un $\gamma = 0$, tad (1.6) vienkāršojas kā (3.6).

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

Sadalot reālajā un imaginārajā daļā kompleksa vektora $\begin{bmatrix} y_1 \\ y_2 \end{bmatrix}$ un kompleksas matricas (3.6) reizinājumu, iegūstam:

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

Iegūtās izteiksmes izmanto rotācijas elementa (*RE*) realizācijai *FPGA*. Vienkāršota *RE* blokshēma, kas realizē (3.7) izteiksmes, parādīta 3.8. attēlā. Katram vienkāršojumam un katrai rotācijas matricas struktūrai nepieciešams izveidot savu *VHDL* kodu, kuru skaits var sasniegt pat vairākus tūkstošus [P12]. Šis uzdevums ir ārkārtīgi darbietilpīgs, tāpēc



3.8. att. Rotācijas elements (ϕ, $\psi = \frac{\pi}{2}$, $\gamma = 0$, 424) [P9]

tika izveidota automatizēta sistēma (skat. 3.6. nodaļu) visu rotācijas matricu struktūru un vienkāršojumu pārbaudei un realizācijai *VHDL* kodā.

9. tabula. Aritmētisko operāciju skaits dažiem parametru variantiem, nelietojot atmiņas elementus [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

Vienkāršojumu izmantošana, protams, samazina nepieciešamo aritmētisko operatoru skaitu. 9. tabulā parādīts kompleksajai rotācijai nepieciešamo reizinājumu un summēšanu skaits (ieskaitot sinusu un kosinusu savstarpējos reizinājumus (3.3)).

Rezumējums par [P9], [P11] un [P12]

Iegūtie rezultāti

- Ir izstrādāts simboliskās matemātikas algoritms (iebūvēts *EGURIT* [P12]), kas uzdotajam leņķu komplektam ļauj atrast visus iespējamos *EGURM* vienkāršojumus un atbilstošo elementāro spektru izteiksmes.

Secinājumi

- Jebkuram/jebkuriem parametriem (ϕ, ψ vai γ) izvēloties konstantu vērtību, iespējams būtiski samazināt vektora rotācijai nepieciešamo reizināšanu un summēšanu skaitu, bet palielinās dažādo *RE* variantu skaits. Tā, piemēram, izvēloties 6 dažādas

vērtības katram parametram (*variable*, 0, $\pi/6$, $\pi/4$, $\pi/3$, $\pi/2$) un ņemot vērā iespējamo *RE* struktūru skaitu, iegūstam, ka dažādo *RE* modifikāciju skaits sasniedz 6880 [P12].

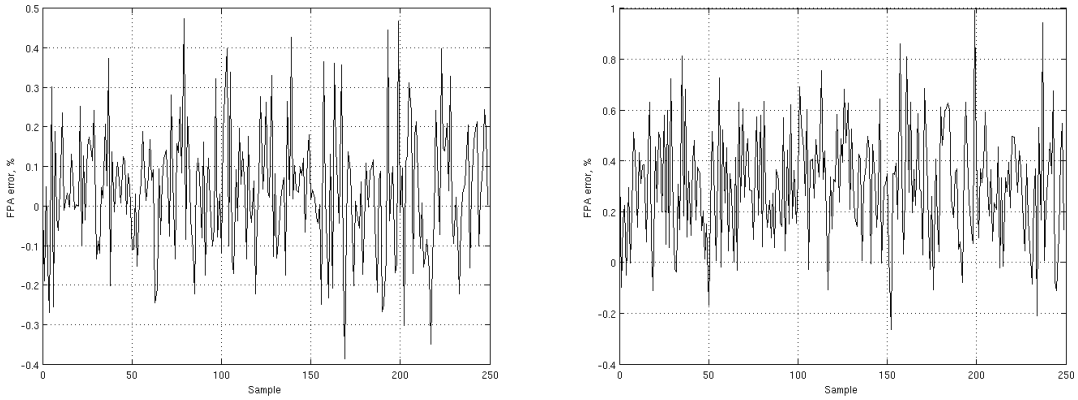
- Konstanto parametru vērtībām, izvēloties lielumus, piemēram, 0, $\frac{\pi}{4}$, $\frac{\pi}{2}$, iegūst dažus (līdz pat 7) tipveida izteiksmju variantus. Viens vai otrs variants atšķiras ar *M*, *A* un *MC* skaitu.

- Lai pārvaldītu (*manage*) iespējamo milzīgo *EGURM* vienkāršojumu un atbilstošo elementāro spektru daudzumu, ir nepieciešams izteiksmju iegūšanas procesu automatizēt.

3.5. Fiksētā punkta kļūda

Šajā apakšnodaļā tiek aplūkota fiksētā punkta kļūda *RE* izejā atkarībā no ieejas signāla un starpsignālu (*RE* iekšienē) nolašu vārda garumiem (*WL*).

Viens no fiksētā punkta kļūdas rašanās iemesliem ir signāla nolašu *WL* samazināšana. *WL* samazināšana ir jāveic, lai algoritmi būtu realizējami reālās fiksēta punkta iekārtās. Atkarībā no fiksētā punkta ieejas, starprezultātu un izejas signālu *WL* mainās arī kļūda, ar kādu tiek iegūts izejas signāls (divu nolašu spektrs). 3.3.1. nodaļā no 7. tabulas redzams, ka mainoties signālu *WL*, mainās arī izmantojamo loģisko elementu skaits.



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

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

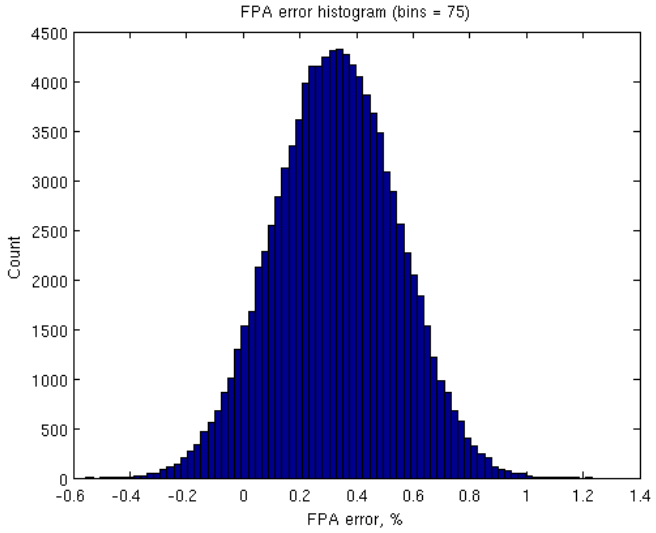
3.9. att. Fiksētā punkta kļūdas 250 nolases garam trokšņveida signālam

3.9. attēlā parādītas 250 nolases gara trokšņveida signāla katras nolases fiksētā punkta kļūdas, ja starprezultātu bitu skaits pēc reizinātāja tiek palielināts par 4 bitiem. Kļūdas attēlotas procentos, attiecībā pret fiksētā punkta dinamisko diapazonu. Kļūdas aprēķināšanai lietota sekojoša formula:

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

kur $\widehat{\mathbf{v}}$ – signāls ar fiksētā punkta kļūdu, $\mathbf{v}$ – signāls bez fiksētā punkta kļūdas (iegūts izmantojot *MatLab* peldošo punktu), $r-$fiksētā punkta diapazons.

FPA kļūdu sadalījuma histogramma, kas iegūta no 100000 nolašu gara trokšņveida signāla, parādīta 3.10. attēlā.



3.10. att. *FPA* kļūdu sadalījuma histogramma ($w_{in} = 8$, $w_{mult} = 8$, $w_{out} = 8$, *floor* noapaļošana)

Dažādiem salīdzinājumiem, daudz ērtāk izmantot vidējo kvadrātisko kļūdu, kuru aprēķina izmantojot sekojošu izteiksmi:

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

Fiksētā punkta kļūdu, protams, ietekmē arī izvēlētais noapaļošanas veids (skat. 3.2.1. nodaļu). 10. tabulā parādītas fiksētā punkta vidējās kvadrātiskās kļūdas pie visiem iespējamajiem noapaļošanas veidiem. Maksimālās *FPA* kļūdu moduļu vērtības pie dažādiem signālu

10. tabula. Fiksētā punkta vidējā kvadrātiskā kļūda ϵ_{MSE}

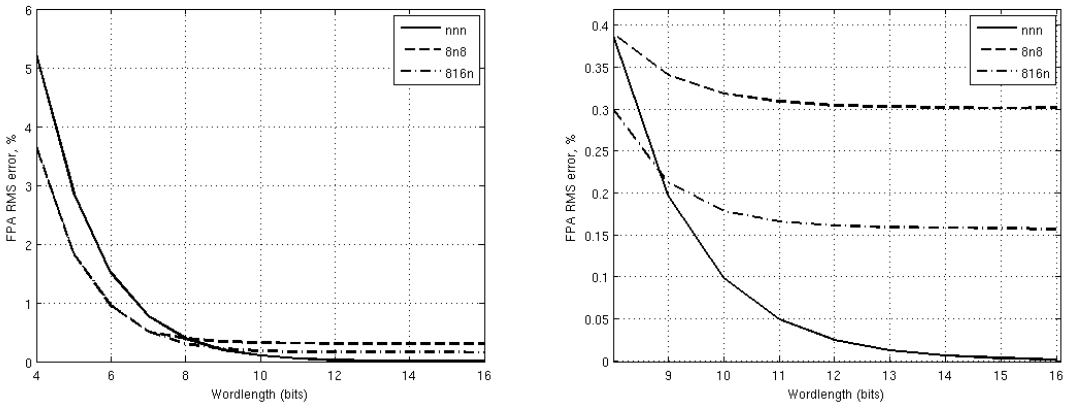
Signal properties			Rounding Type			
w_{in}	w_{mult}	w_{out}	Floor	Fix	Round	Nearest
8	8	8	0.39%	0.40%	0.21%	0.22%
8	16	8	0.30%	0.32%	0.19%	0.19%
8	16	16	0.16%	0.16%	0.15%	0.15%

11. tabula. Fiksētā punkta maksimālās kļūdas modulis $max(abs(\epsilon_{\%}))$

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%

WL, apkopotas 11. tabulā. Vidējās kvadrātiskās kļūdas rēķināšanai un maksimālās kļūdas moduļa noteikšanai izmantotas 100000 nolases garas datu virknes, kas ģenerētas, izmantojot *MatLab*'ā iebūvēto trokšņa ģeneratoru ar *mrg32k3a* algoritmu. Maksimālo *FPA* kļūdu moduļu vērtības pie dažādiem signālu *WL*, apkopotas 11. tabulā.

FPA kļūdu izmaiņas dažādos veidos mainot signālu *WL*, parādītas 3.11. attēlā. Ar "nnn" ("$w_{in}w_{mult}w_{out}$") jāsaprot, ka vienādi tiek mainīti ieejas (w_{in}), signāla pēc reizinātāja (w_{mult}) un izejas (w_{out}) signālu *WL*. "8n8" nozīmē, ka ieejas un izejas signālu *WL* ir



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

3.11. att. Fiksētā punkta vidējās kvadrātiskās kļūdas, mainot *WL* (*floor* noapaļošana)

8 biti, bet mainīts tiek signāla pēc reizinātāja *WL* un, attiecīgi "816n" – ieejas signāla *WL* ir 8, bet pēc reizinātāja 16 biti un mainīts tiek izejas signāla *WL*.

Rezumējums (rezultāti nav publicēti)

Lai gan šajā apakšnodaļā aprakstītie rezultāti (līknes, tabulas, formulas) nav publicēti, izstrādātais kļūdu novērtēšanas rīks ir iebūvēts *EGURIT* ([P12]).

Iegūtie rezultāti

- Ir izstrādāts *RE FPA* izraisītās kļūdas novērtēšanas rīks.

Secinājumi

- Izveidotais rīks ļauj atrast rotācijas rezultātā radušās vidēji kvadrātiskās kļūdas pie dažādiem:
  - ieejas signālu, starpsignālu (pēc reizinātājiem) un izejas signālu nolašu vārda garumiem,
  - noapaļošanas veidiem.
- Izveidoto rīku ir nepieciešams (paredzēts) uzlabot tā, lai tas ļautu automātiski noteikt optimālos vārda garumus un noapaļošanas veidu pie uzdotās kļūdas.

3.6. *RE* realizācijas automatizācija [P11], [P12]

Kā minēts 1.2.1. nodaļā, vispārinātās elementārās rotācijas matricas struktūru skaits vienāds ar 32, ja ϕ,ψ,$\gamma \in [0,90°]$. Ņemot vērā iespējamos vienkāršojumus (modifikācijas) (skat. 3.4. nodaļu), dažādo iespējamo *RE* realizāciju skaits var sasniegt vairākus tūkstošus, kas pamatīgi apgrūtina manuālu *RE FPGA* realizāciju. Acīmredzama kļūst automatizētas *RE* realizācijas sistēmas izveides nepieciešamība.

Automatizācija tiek veikta izmantojot *MatLab/Simulink*, *Quartus II* un *ModelSim* programmas. Minētās programmas tiek lietotas sekojošu uzdevumu veikšanai:

- *MatLab* – Automatizācijas vadība, rotācijas izteiksmju unitaritātes pārbaude, to sadalīšana reālajā un imaginārajā daļās, izmantojot *Symbolic Math Toolbox*(*SMT*). Rezultātu iegūšana izmantojot peldošā punkta aritmētiku (*double precision*) un iegūto izteiksmju pārveidošana fiksētā punkta aritmētikā ar ***fi()*** objektu.
- *Simulink* – Sistēmas simulēšana fiksētā punkta aritmētikā, *VHDL* koda ģenerēšana izmantojot *HDL coder*,
- *ModelSim* – *VHDL* koda simulēšana,
- *Quartus II* – *VHDL* koda kompilācija izvēlētajai *FPGA* mikroshēmai, *FPGA* programmēšana.

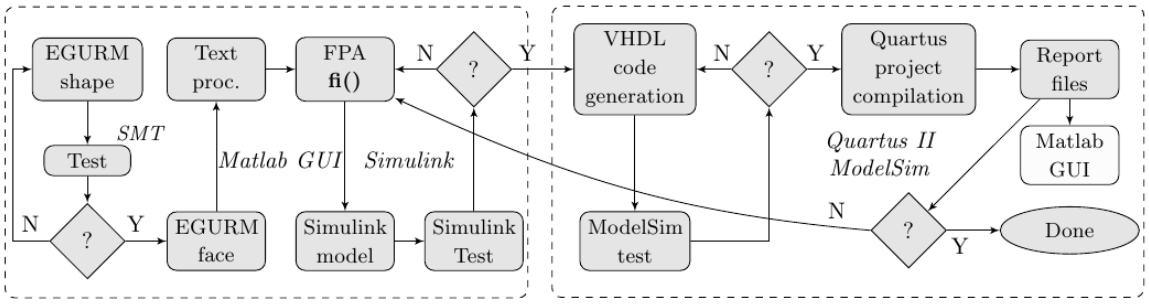
MatLab/Simulink tiek uzskatīta par standarta inženiermatemātisko programmēšanas valodu sarežģītu *DSP* algoritmu izstrādei. *Quartus II* izvēle, galvenokārt, saistīta ar pieredzi darbā ar to un pieejamiem izstrādes līdzekļiem (*Development kits*), kas iegādāti darbojoties vairākos projektos. *Mentor Graphics ModelSim* ir *Altera*s rekomendētā *HDL* kodu simulēšanas vide. Gan *Quartus II*, gan *ModelSim* var palaist komandrindu režīmā, kas ļauj veikt *ModelSim* simulēšanu un *Quartus II* projekta kompilēšanu *MatLab*ā.

Visu automatizācijas procesu var sadalīt vairākos posmos (sīkāks apraksts sniegts 3.6.1. nodaļā):

- Darbs ar simbolisko matemātiku,
- Fiksētā punkta aritmētikas (*FPA*) *WL* izvēle,
- *MatLab* funkcijas ģenerācija,
- *HDL* koda ģenerācija,
- Testēšana.

3.6.1. Automatizācijas posmi

Automatizētās sistēmas vienkāršota blokshēma parādīta 3.12. attēlā, kur ar *EGURM shape* jāsaprot rotācijas matricas struktūras izvēle, ar *EGURM face* – vienkāršojuma izvēle, *Text proc.* – standarta matemātisko izteiksmju teksta apstrāde, bet *FPA* ***fi()*** – *MatLab* **fi()** objekta (fiksētā punkta aritmētikas objekts *MatLab*-ā) pielietošana iegūtajām izteiksmēm.



3.12. att. Automatizētās sistēmas vienkāršota blokshēma [P12]

Darbs ar simbolisko matemātiku Kā pirmais, jāpārbauda visu iespējamo kompleksо rotācijas matricu (1.3) struktūru (izstrādātajos līdzekļos (skat. 3.13. att.) apzīmēts ar "*Shape*") ortogonalitāte (jeb unitaritāte). Ortogonālās struktūras saglabā datu bāzē. Pēc tam, jāizvēlas kāds no vienkāršojumiem (vienkāršojumu apzīmēšanai angļu val., tiek ieviests termins "*Face*"(skat. 3.13. att. 69. lpp)), piemēram, $(\phi, \psi = \frac{\pi}{4}, \gamma)$. Ar *MatLab* funkciju **latex()**, izteiksmes iegūst attēlošanai LaTeX formā.

Pēc tam no kompleksajām izteiksmēm, līdzīgi kā (3.7), jāiegūst reālās izteiksmes. Kā pēdējais solis darbā ar simbolisko matemātiku – reālās izteiksmes jāsadala starprezultātu izteiksmēs tā, lai katrā no izteiksmēm būtu tikai reizinātāji vai tikai summatori. Tā, piemēram, sadalot (3.7), $y_{1\,Re}$ iegūst ar sekojošām izteiksmēm:

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

Fiksētā punkta aritmētikas (*FPA*) *WL* izvēle Šajā solī, izmantojot izveidoto *WL* noteikšanas algoritmu vai brīvi ierakstot, tiek izvēlēti *WL* visiem starprezultātiem. Lietojot *fi*() objektu, svarīgi ir norādīt gan kopējo *WL*, gan arī daļskaitļa (*fraction*) *WL*. Jāpatur prātā, ka ar algoritmu tiek veikta ieejas signāla vektora rotācija, tāpēc, ja ieejas signāls ir *Q1.x* fiksētā punkta aritmētikas formātā, neveicot ieejas signāla vektora garuma ierobežojumus, izejas signāls var iziet ārpus *Q1.x* aritmētikas diapazona (piemēram, vektoru $[-1,\ -1]$ pagriežot par 45° pulksteņrādītāja virzienā, tiek iegūts vektors $[0,\ -\sqrt{2}]$). Šī iemesla dēļ, izejas signāls tiek aprakstīts *Q2.x* aritmētikā, kas nodrošina fiksētā punkta diapazonu $[-2,\ 2-\frac{1}{2^x}]$. Starprezultātos, atkarībā no summatoru skaita, korektai darbībai var būt nepieciešami arī citi fiksētā punkta aritmētikas varianti (*Q3.x*, *Q4.x*).

Autora izveidotā *WL* aprēķina algoritms darbojas pēc sekojoša principa:

- tiek uzstādīti izvēlētie *WL* starprezultātiem pēc reizinātājiem (w_{mult}), saglabājot ieejas signāla nolasēm *Q1.x* aritmētiku,
- signālu reizinot ar konstanti, signāla nolašu *WL*, kā arī veselās/daļskaitļa daļas bitu skaita attiecība nemainās,
- summatoru izejā tiek palielināts veselās daļas (*integer*) *WL*, nemainot daļveida daļas (*fraction*) *WL*.

MatLab* funkcijas ģenerācija** No iegūtajām simboliskajām izteiksmēm un *FPA WL*, jāizveido *MatLab* funkcija, kura tiks izmantota *HDL* koda ģenerācijai. Lietotais variants – *HDL* koda ģenerācija, izmantojot *Embedded MatLab Function* (*EMLF*), ir daudz efektīvāks par otro iespējamo variantu – *Simulink* modeļa izveidi, izmantojot *MatLab* funkciju ***add_block(). *MatLab* funkciju var iekļaut *Simulink EMLF*, ja tās pirmajā rindiņā ievieto speciālo norādi – **%#eml**.

***HDL* koda ģenerācija** *Simulink* modelī tiek ievietota *EMLF*, kurā ir norāde uz izveidoto *MatLab* funkciju. *EMLF* jāpievieno visi izvadi (*X1Re*, *X1Im*, *cf*, utt.) un jānorāda ieejas signālu *Q1.x FPA WL*. Simboliskās izteiksmes ir izveidotas tā, lai tās saturētu tikai reālās reizināšanas un summēšanas, tāpēc, kā ieejas parametri ir arī $\sin(\phi)$, $\cos(\phi)$, $\sin(\psi)$, utt. Lai pagriezēja modulis saturētu pēc iespējas mazāk reizinātājus, kā ieejas parametri ir arī nepieciešamo sin / cos reizinājumi, tādi kā, piemēram, $ccg = \cos(\phi) \cdot \cos(\gamma)$. *HDL* kods tiek ģenerēts izsaucot *MatLab* funkciju **makehdl()**, kuru attiecīgi nokonfigurējot tiek uzģenerēti arī *Quartus II* un *ModelSim* skriptu faili. Par to sīkāk, nākamajā, 3.6.2. nodaļā.

Testēšana Testēšanā pavisam tiek izmantoti un salīdzināti 3 dažādi testu varianti:

- *MatLab* izteiksmju peldošā punkta tests,
- *Simulink* ar fiksēto punktu tests,

- *HDL* koda *ModelSim* tests.

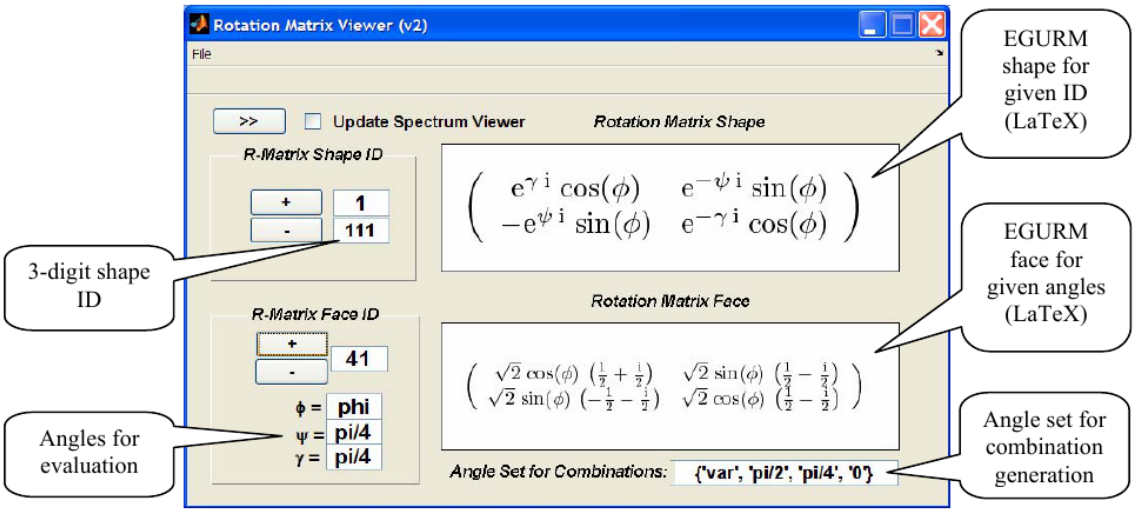
Visiem testiem, visi ieejas signāli tiek ģenerēti izmantojot *MatLab* trokšņa ģeneratoru (ar vienmērīgu sadalījumu), ar *mrg32k3a* algoritmu, kas ļauj ģenerēt vairākas trokšņa virknes ar aptuveno periodu 2^{127} nolases. Iegūtās ieejas signālu virknes noformē katram no testiem – kā *MatLab* vektorus peldošā punkta testam, kā struktūru *MatLab* darba atmiņā *Simulink* testam un kā *.do failu *ModelSim* testam. Peldošā punkta un *Simulink* rezultātus izmanto vidējās kvadrātiskās kļūdas (*MSE*) aprēķiniem. *ModelSim* iegūtos rezultātus salīdzina ar *Simulink* rezultātiem, lai pārliecinātos, ka uzģenerētais *VHDL* kods darbojas pareizi.

3.6.2. Automatizācijas veikšanai izstrādātie līdzekļi

Zemāk uzskaitīti automatizācijas līdzekļi (*MatLab* programmas) un to galvenās funkcijas.

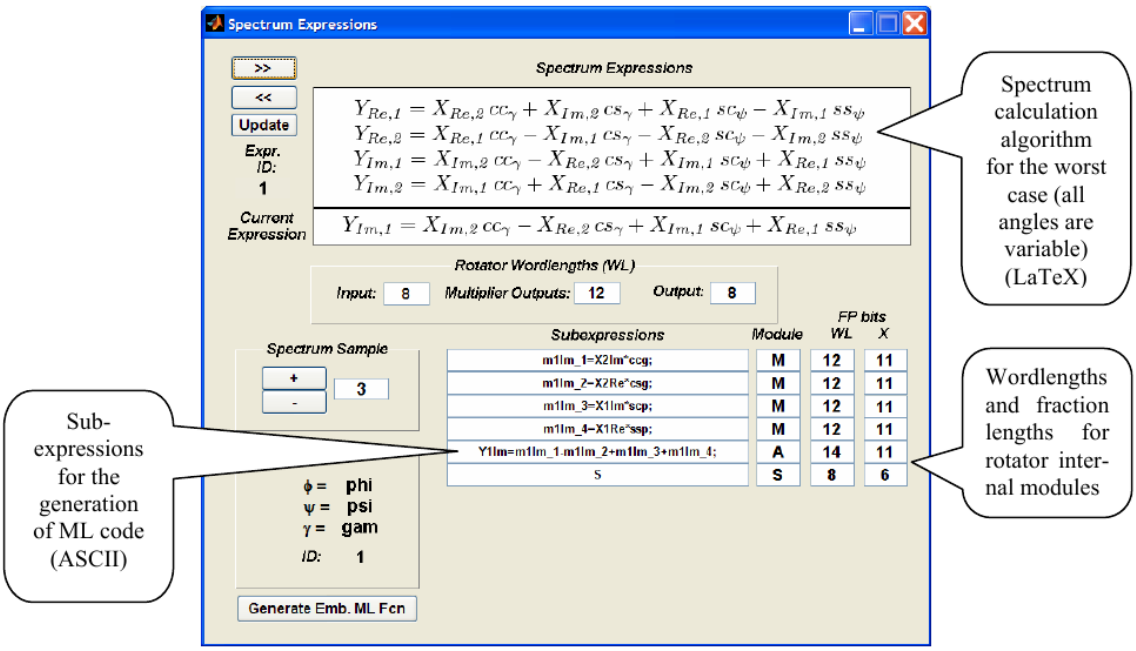
Rotation matrix viewer (skat. 3.13. att.) (∼ 750 *MatLab* rindas)

- Rotācijas matricu struktūru (*shape*) izvēle
- Rotācijas matricu vienkāršojumu (*face*) izvēle
- Vienkāršojumu parametru kombināciju izvēle
- Simbolisko izteiksmju LaTeXattēlojums



3.13. att. Rotation Matrix Viewer dialoglogs [P12]

Šo līdzekli izmanto, lai no iepriekš sagatavotām elementāro rotācijas matricu struktūrām izvēlētos vienu, kuru turpmāk realizēt *FPGA* mikroshēmās. Šo pašu līdzekli izmanto, lai izvēlētajai struktūrai piekārtotu kādus vienkāršojumus, ja tie nepieciešami.



3.14. att. *Spectrum Expressions* dialoglogs [P12]

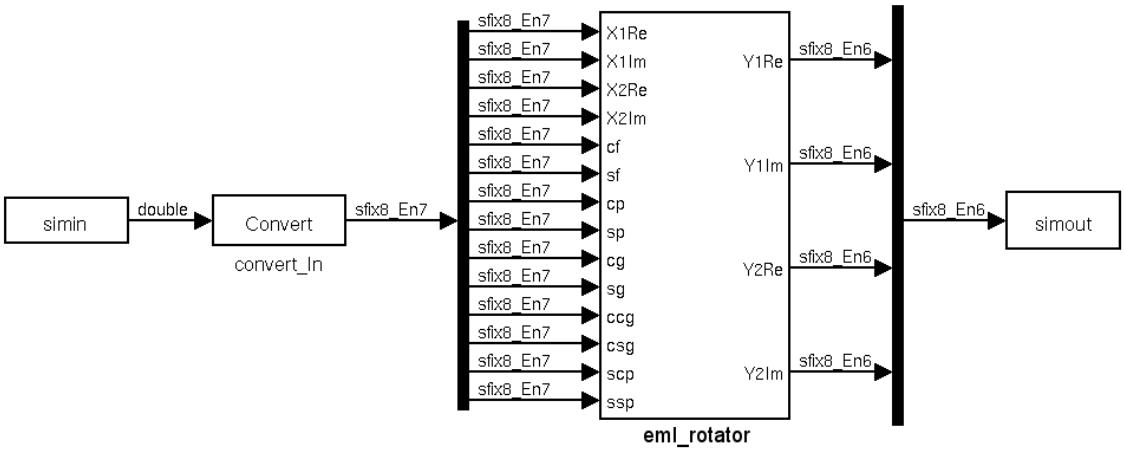
Spectrum Expressions (skat. 3.14. att.) (∼ 2100 *MatLab* rindas)

- Spektra kalkulācijas izteiksmju LaTeXattēlojums
- Spektra koeficienta izvēle un aprēķināšanas *ASCII* izteiksmes
- Automātiska/manuāla signālu *WL* uzstādīšana
- *Embedded MatLab function* ģenerācija

Ar šo līdzekli veic fiksētā punkta aritmētikas *WL* uzstādīšanu un *MatLab* funkcijas ģenerāciju. Uzģenerēto funkciju izmanto 3.15. attēlā redzamais *Simulink* modelis.

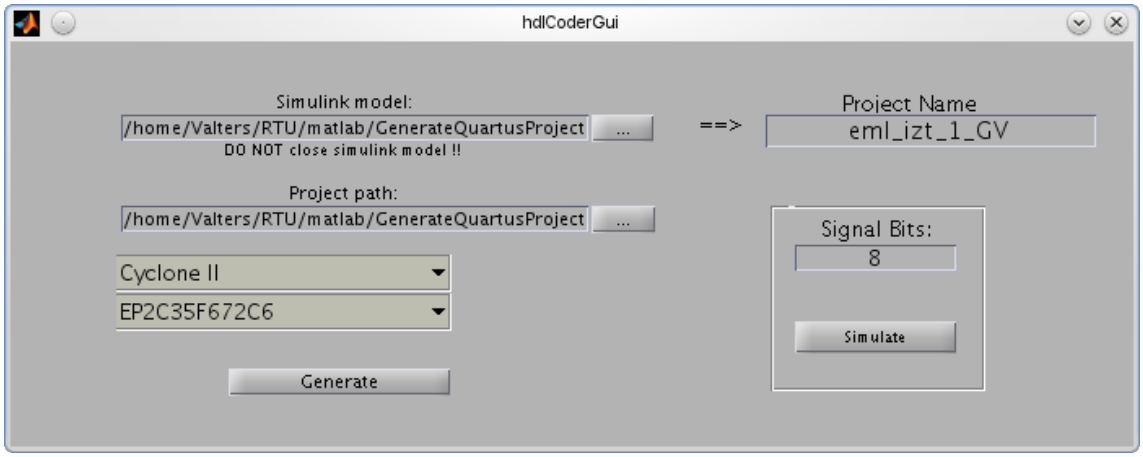
HDLCoderGUI (skat. 3.16. att.) (∼ 600 *MatLab* rindas)

- Peldošā punkta un *Simulink* tests



3.15. att. *Simulink* modelis *VHDL* koda ģenerācijai [P12]

- *Simulink HDL* kodera konfigurācija
- *VHDL* koda un *Quartus II* un *ModelSim* skriptu ģenerācija
- *Quartus II* projekta apakškatalogu sagatavošana
- *ModelSim* tests un rezultātu salīdzināšana ar *Simulink* rezultātiem
- *Quartus II* kompilācija
- Rezultātu attēlošana



3.16. att. *HDLcoderGUI* dialoglogs

12. tabula. Kompilācijas/simulācijas laiks (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

Pēc *HDL* kodera konfigurācijas, automātiski tiek uzģenerēts *Quartus* skriptu fails *projNosaukums_quartus.tcl* un *ModelSim* skripta fails *projNosaukums_compile.do*. Abi šie skriptu faili nepieciešami, lai komandrindu režīmā varētu veikt, attiecīgi, *Quartus* projekta izveidi un kompilāciju un *ModelSim* projekta izveidi un kompilāciju. Pēc *ModelSim* kompilācijas var veikt simulāciju. Simulācijas rezultāti tiek saglabāti **.lst* failā.

Iespēja startēt *Quartus* un *ModelSim* komandrindu režīmā, ļauj veikt attiecīgās simulācijas un projekta kompilāciju, jo *MatLab*ā var palaist jebkuru sistēmas programmu, izmantojot funkciju **system()**. Pirms tam, katru reizi tiek izveidots katalogs, kura nosaukums satur datuma un laika informāciju.

12. tabulā parādīts aptuvenais laika patēriņš, kāds nepieciešams viena projekta simulācijai un kompilācijai. Veicot *RE* realizācijas automatizāciju, īpaša uzmanība jāpievērš *Quartus* projekta kompilācijas ilgumam–tas ir stipri atkarīgs no izvēlētā *FPGA* parametriem, tāpēc, sistēmas izstrādes sākumā, novērtējot *FPA* kļūdas, nav lietderīgi izvēlēties ļoti jaudīgus *FPGA* (piemēram, *Stratix IV E820*).

Rezumējums par [P11] un [P12]

Iegūtie rezultāti

- Ir izanalizēts stāvoklis automatizēto līdzekļu izveidē, kas ir orientēti uz *VHDL* koda sintēzi un saistīti ar Jakobi rotācijām.
- Ir izveidots automatizēts līdzeklis *EGURIT* elementārās vispārinātās kompleksās rotācijas *VHDL* koda ģenerācijai un sintēzei *Alteras FPGA*. *EGURIT* izmanto šādas programmatūras:

- *MatLab/Simulink*,
- *Altera Quartus II*,
- *Mentor Graphics ModelSim*.

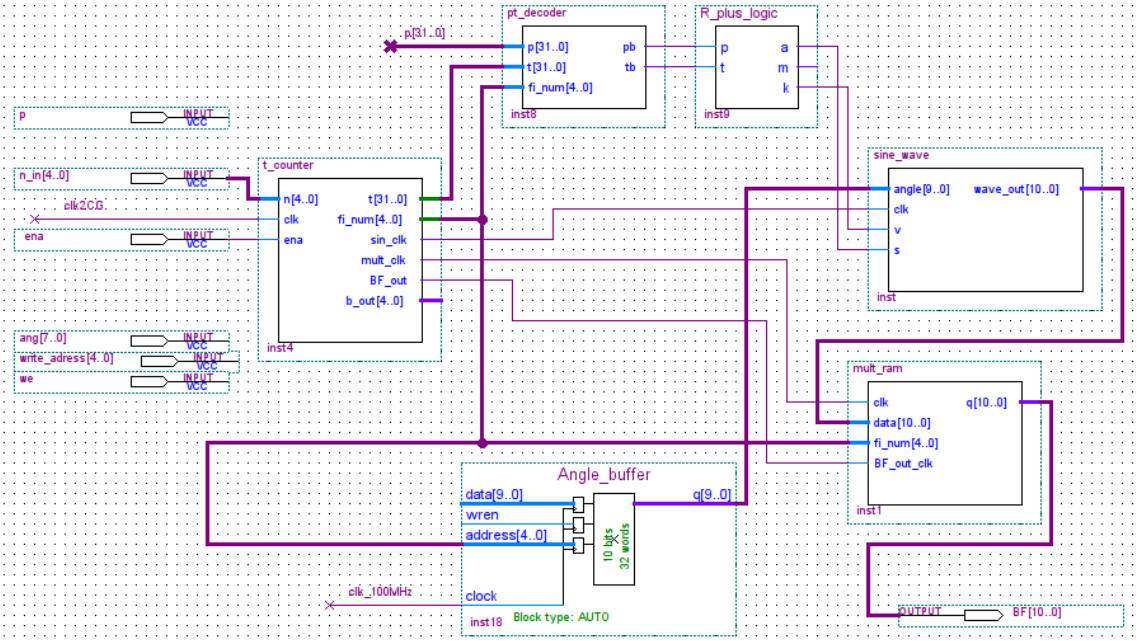
Secinājumi

- Pieejamajā literatūrā nav atrasti izveidotā *EGURIT* analogi, kas ir orientēti uz Jakobi rotācijas vai tās paveidu *VHDL* koda sintēzi.
- Pieejamās literatūras analīze rāda, ka *Matlab/Simulink* vide ir visizplatītākā vide tādu automatizēto sistēmu izveidei, kas ir saistītas ar signālu ciparapstrādi.
- Pieejamās literatūras analīze rāda, ka visbiežāk signālu ciparapstrādei tiek izmantotas *Altera* un *Xilinx FPGA* mikroshēmas un *VHDL* kodi tiek sintezēti tieši šo firmu *FPGA* saimēm.
- *Quartus II* un *ModelSim* komandrindu režīms un *MatLab* iespējas ļauj veikt efektīvu automatizācijas līdzekļu izstrādi.
- Ar izstrādāto automatizācijas līdzekli, relatīvi īsā laikā iespējams sintezēt un realizēt *FPGA VHDL* kodus, jebkurai no tūkstošiem iespējamo *RE* struktūru un modifikāciju (skat. 3.4. nodaļas secinājumus vai [P12]) un noteikt konkrēta *FPGA* izmantotos resursus (*LE*, *DSP* elementu skaitu un laika parametrus).
- *EGURIT* relatīvi īsā laikā ļauj iegūt:
  - milzīga skaita dažādo *RE* struktūru parametrus,
  - ļauj novērtēt arī uz rotācijas leņķiem bāzētu pārveidojumu *FPGA* īstenošanas iespējas pie $N > 2$ (novērtēt potenciāli izmantojamos resursus – *LE*, *DSP* elementu skaitu un laikus).
- Izstrādātā līdzekļa ātrdarbību nosaka galvenokārt *Quartus II* projekta kompilācijas laiks (no dažām sekundēm līdz vairākām minūtēm), kas, galvenokārt, ir atkarīgs no *FPGA* čipa sarežģītības un datora ātrdarbības, kādēļ resursu pietiekamības novērtēšanas stadijā vēlams ir izmantot čipus ar pēc iespējas mazāku *LE* skaitu.
- Izstrādāto automatizācijas sistēmu ir iespējams adaptēt arī citu firmu *FPGA* izstrādes līdzekļiem, piemēram, *Xilinx ISE*, *Synopsis*, u.c. (tiem izstrādes līdzekļiem, kuriem ir iespējama tā saucamā skriptēšana (*scripting*)).

4. Eksperimentālās ierīces

4.1. Eksperimentālā CRAIMOT funkciju ģeneratora realizācija FPGA

Informācija par **CRAIMOT** *BF* ģeneratoru atrodama [P1] un [P4]. 2.1. attēlā redzamo ģeneratoru varam realizēt *FPGA* izmantojot *Quartus II* programmatūru (skat. 4.1. att.). Turpmākajās izstrādēs *Quartus II* grafiskā vide vairs netiek izmantota, jo visas sistēmas programmēšana *VHDL* valodā ir efektīvāka (arī attēlotajā blokshēmā, visi bloki ir rakstīti *VHDL* ($\approx$ 300 rindas), bet grafiskā vide izmantota tikai bloku savstarpējai savienošanai) un sākot ar *Quartus II ver. 10.0* netiek atbalstīta simulēšana grafiskajā vidē.



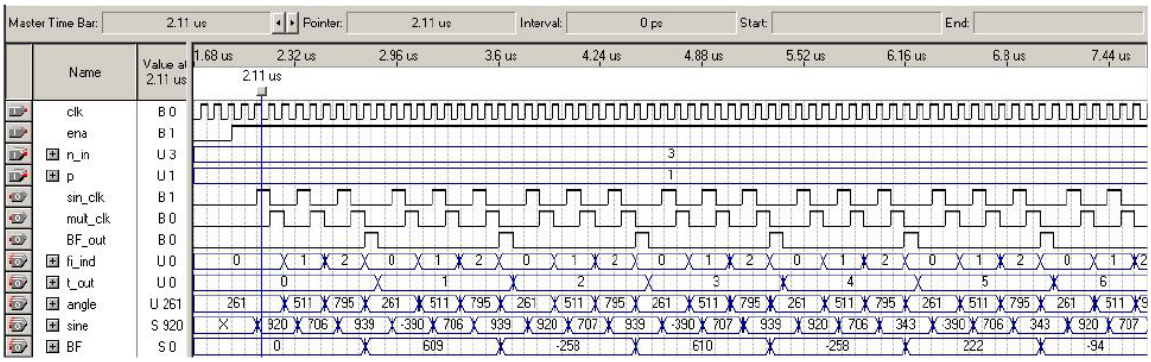
4.1. att. **CRAIMOT** *BF* ģenerators realizēts *Quartus II ver. 6.0* grafiskajā vidē

Izveidotā ģeneratora *FPGA* realizācijas parametri apkopoti 13. tabulā.

BF-u ģeneratora simulācijas laika diagrammas redzamas 4.2. attēlā. Lai demonstrētu

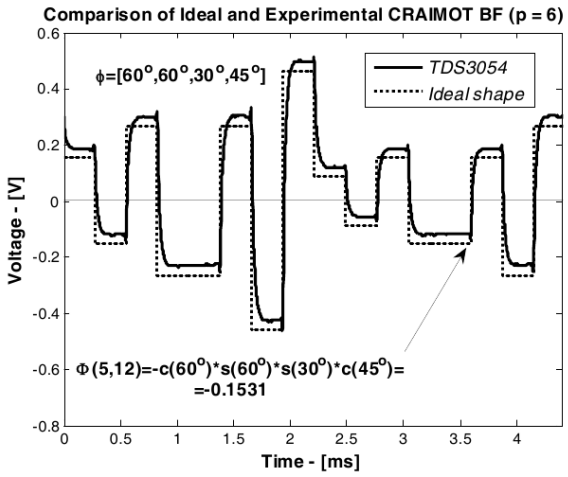
13. tabula. *BF*-ju virknes arhitektūras ģeneratora parametri [P1]

Parametrs	Vērtība
Arējā takts impulsu ģeneratora frekvence	$f_{clk} = 48$ MHz
Maksimālais leņķu skaits	$n = 32$
Maksimālais *BF* garums	$N = 2^n = 4294967296$
BF WL	$w = 10$ biti, $Q1.x$ *FPA*
BF nolašu ilgums	$T_{s\,BF} = (3 \cdot n + 1)/(f_{clk}/4)$
BF atkārtošanās periods	$T_{BF} = 1/(N \cdot T_s)$



4.2. att. **CRAIMOT** *BF*-u ģeneratora simulācijas laika diagramma izmantojot *Quartus II ver. 6.0* [P1]

ģeneratora darbību, tika izmantots *FPGA* izstrādes rīka audio kodeks, ciparu signāls tika pārveidots par analogu signālu un attēlots uz osciloskopa ekrāna. 4.3. attēlā, importētie dati no osciloskopa ekrāna salīdzināti ar *MatLab* ģenerētajiem datiem.



4.3. att. Ideālā un experimentālā *BF*-a ($N = 16, p = 6$) (uzskatāmības dēļ ir piesummēta līdzkomponente) [P1]

Aprakstītais virknes arhitektūras [P1] **CRAIMOT** *BF* ģenerators ir viena no pirmajām praktiski realizētajām uz leņķiem bāzētajām eksperimentālajām iekārtām.

Vienlaicīgi ar virknes arhitektūras *BF* ģeneratora izstrādi tika izveidots arī paralēlās

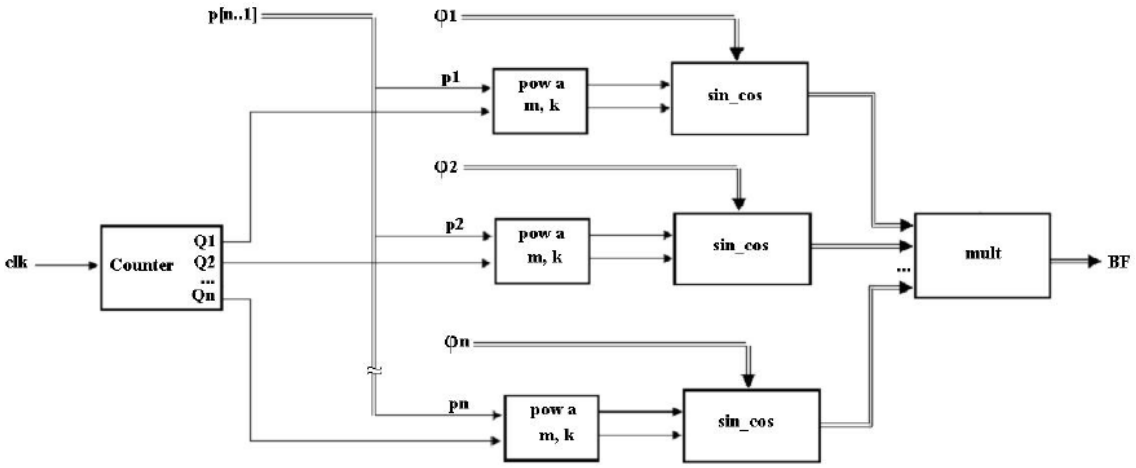
14. tabula. *sin/cos* komponenta izmantojamo resursu skaits ([P4], uzlabots)

	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	$n \cdot 2^{n_\alpha} \cdot w$

15. tabula. *BF*-ju paralēlās arhitektūras ģeneratora parametri [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	$T_{BF} = 1/(N \cdot T_s)$

arhitektūras *BF* ģenerators [P4]. Abu arhitektūru sastāvā ietilpst *sin/cos* vērtību formēšanas komponents.No vērtību formēšanas veida (lineārā interpolācija, iekšējā atmiņa, *CORDIC*) ir atkarīgas *BF*-u iegūšanas kļūdas un izmantojamo resursu daudzums (skat. 14. tabulu). Paaugstinot profesionālo kompetences līmeni, *VHDL* kodi nepārtraukti tiek uzlaboti, tāpēc 14. tabulas dati nesakrīt ar [P4] publicētajiem. Veidojot *BF* ģeneratoru paralēlajā arhitektūrā, vairāk tiek izmantota *FPGA* priekšrocība – paralēli elementi, kas nodrošina augstāku ātrdarbību (skat. 15. tabulu), salīdzinot ar virknes arhitektūru. Izmantojamo *LE* skaits no *BF* garuma ir atkarīgs, jo katras *BF* nolases ģenerēšanai nepieciešami paralēli $n = \log_2(N)$ *sin/cos* bloki (skat. 4.4. att.).



4.4. att. **CRAIMOT** *BF* ģeneratora paralēlās arhitektūras vienkāršota blokshēma [P4]

Rezumējums par [P2] un [P5]

Iegūtie rezultāti

- *FPGA* ir izveidotas divas **CRAIMOT** *BF* ģeneratora versijas.
- Ir iegūta izteiksme, kas ļauj noteikt *BF* vienas nolases minimālo ilgumu uz *EP1C12* realizēta čipa pie takts frekvences 48 MHz.

- Ir iegūts resursu novērtējums trīs dažādiem alternatīviem paņēmieniem *sin/cos* vērtību iegūšanai:
  - ar CORDIC algoritmu,
  - ar optimizētās lineārās interpolācijas (*LININT*) paņēmienu,
  - ar *RAM* tabulu.

Secinājumi

- Ģeneratora virknes arhitektūra ļauj ģenerēt ļoti garas *BF*-as ($N = 2^{32}$) pie salīdzinoši neliela aizņemto *LE* skaita.
- Vienas nolases minimālais ilgums mainās lineāri atkarībā no leņķu skaita (n). Tā, piemēram, leņķu skaitam mainoties robežās no 5 līdz 12, vienas nolases minimālais ilgums mainās no aptuveni vienas līdz trijām mikrosekundēm.
- *BF* garums neietekmē izmantojamo *LE* skaitu, ja *BF* ģenerators realizēts, izmantojot virknes arhitektūru.
- Ģeneratora ar virknes arhitektūra ātrdarbība ir vairāk nekā pietiekama audio signālu sintēzei.
- Realizējot *BF* ģeneratoru paralēlajā arhitektūrā [P4], *BF* nolašu iegūšanas frekvence ir vismaz piecas reizes augstāka, nekā realizējot virknes arhitektūrā [P1].
- Izmantojamo *LE* skaits, *BF* ģeneratoru realizējot paralēlajā arhitektūrā, ir lineāri atkarīgs no leņķu skaita (vai logaritmiski no *BF* garuma) un *sin/cos* vērtību iegūšanas veida, tā, piemēram, *sin/cos* vērtību iegūšanai izvēloties *CORDIC* algoritmu, izmantojamo *LE* skaits vienāds ar $n \cdot 372$.
- Ir konstatēts, ka abām ģeneratora arhitektūrām, *sin/cos* komponente(s) (bloks(i)) aizņem lielāko ģeneratoram izmantojamo *FPGA* resursu daļu.
- Attīstoties tehnoloģijai un arvien palielinoties uz *FPGA* esošo *RAM* bitu skaitam, par visefektīvāko tiek atzīta *RAM* tabulas izmantošana *sin/cos* vērtību iegūšanai, jo šādā gadījumā netiek izmantoti *LE* un *DSP bloki*.
- Optimizētās lineārās interpolācijas algoritms *sin/cos* vērtību iegūšanai ir lietderīgs tikai tad, ja uz čipa "ir lieki" reizinātāji.

4.2. Eksperimentālā signālu spektra analizatora realizācija FPGA

4.2.1. Virknes arhitektūras signālu spektra analizators [P2]

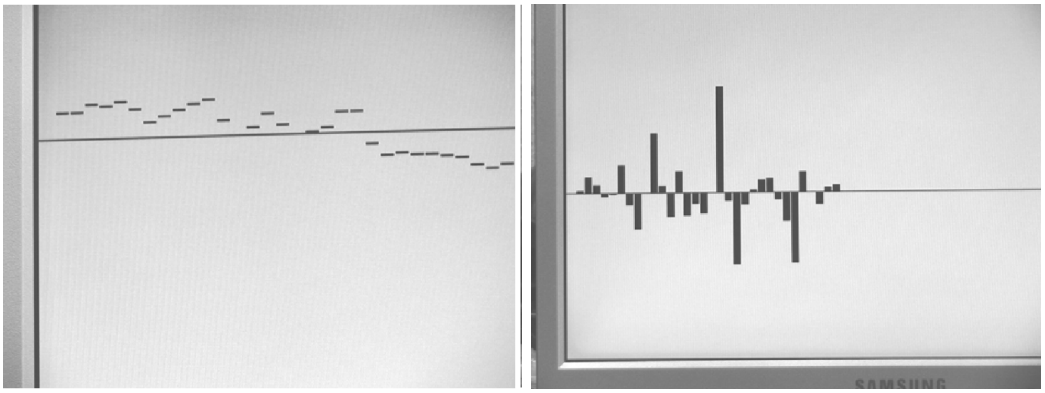
Informācija par signālu **CRAIMOT** spektra analizatora realizāciju, kurā tiek izmantots *BF* ģenerators, publicēta [P2]. 16. tabulā parādīti uz *BF*-u ģeneratora balstīta reāla laika signālu analizatora darbības parametri.

16. tabula. Analizatora parametri [P2]

Parametrs	Vērtība
Arējā takts impulsu ģeneratora frekvence	$f_{clk} = 48$ MHz
Maksimālais signāla garums: •reālā laikā: •datu savākšanas režīmā:	 Ja $N = 32$, tad $F_s = 44.1kHz$ Ja $N = 64$, tad $F_s = 8.2kHz$ $N = 1024$
Signāla un spektra *WL*	$w = 10$ biti, $Q1.x$ *FPA*

Uz *FPGA* izveidotajam analizatoram tika pievienots vienkāršs *VGA* video kontrolieris signāla un iegūtā **CRAIMOT** spektra attēlošanai uz *LCD* ekrāna. 4.5. attēlā redzamas ekrāna fotogrāfijas.

Izveidotais 1-D signālu spektra analizators ir pirmā šāda veida *FPGA* realizētā eksperimentālā iekārta, kas devusi augsni turpmāku signālu apstrādes iekārtu izstrādei, izmantojot efektīvākus algoritmus (piemēram, izmantojot *RE*).



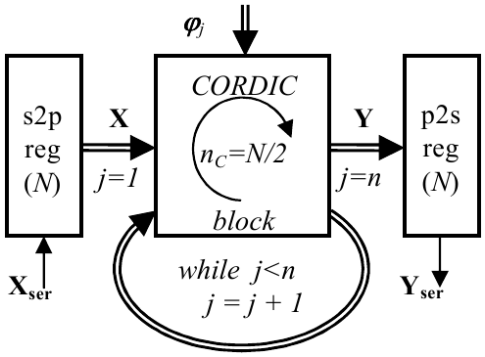
(a) Signālu spektra analizatora ieejas signāla daļa

(b) Signāla **CRAIMOT** spektrs

4.5. att. LCD ekrāna momentuzņēmums [P2]

4.2.2. Uz *RE* balstīts signālu spektra analizators-sintezators [P5]

Šis signālu spektra analizators, līdzīgi kā iepriekšējā apakšnodaļā aprakstītais, veic signālu spektra aprēķinu paralēliem datiem. 4.6. attēlā parādīta vienkāršota signāla spektra aprēķina blokshēma. Ar "*CORDIC block*" jāsaprot *RE* masīvs, kura sastāvā esošo *COR-*



4.6. att. Vienkāršota **RA-HT** spektra analizatora blokshēma [P5]

DIC RE skaitu nosaka izvēlētā analizatora-sintezatora arhitektūras versija (skat. 17. tabulu), ko nosaka *FOR* cilpu skaits. Cilpu lietošana

- atvieglo fī-pārveidojuma īstenošanu, jo, piemēram, versijai 1 (no 17. tabulas) spektra iegūšanai nepieciešams tikai viens takts impulss,
- palielina izmantojamo *LE* skaitu, jo pārveidojums tiek izvērsts kā paralēla *FPGA* struktūra.

18. tabulā apkopoti uz *RE* balstīta signālu analizatora-sintezatora darbības parametri.

Rezumējums

Iegūtie rezultāti

- Ir izveidots pirmais eksperimentālais, uz *FPGA* balstīts, virknes arhitektūras 1-D signālu **CRAIMOT** spektra analizators.
- Ir izveidotas *FPGA* trīs, uz *RE* balstītas (paralēlās arhitektūras), (publikācijā [P5] *RE* masīvs apzīmēts kā *CORDIC block*) signālu **RA-HT** spektra analizatora versijas.

17. tabula. Analizatora/sintezatora versiju salīdzinājums [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	$n_C = 1$	$\approx 700 + 600 \cdot n_C + 50 \cdot n \cdot 2^{n-1}$	$\approx 70 \cdot n \cdot 2^{n-1}$

18. tabula. *SANSYN* moduļa parametri [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	$T_s = n \cdot T_{CORDIC}$

Secinājumi

- Uz *BF* ģeneratora balstīta (ar virknes arhitektūru) signālu spektra analizatora ātrdarbība ir zema un praktiski noderīga var būt telefonijas ($Fs = 8$kHz) un biomedicīnisko (piemēram, *EEG*) signālu ekspresanalīzei.

- Izveidotā virknes arhitektūras analizatora darbība ir izvērsta laikā (skat. 2.2.2. nodaļu), tāpēc analizatora realizācijai ir nepieciešami salīdzinošo maz *LE* un analizators ir piemērots ierīcēm ar mazu jaudas patēriņu.

- Tā kā uz rotācijas elementiem balstītā analizatora versija ir izvērsta *FPGA* paralēlā struktūrā, tad uz *RE* izveidotais signālu spektra analizators ir daudz ātrdarbīgāka (≈ 65 reizes, ja $n = 5$) nekā virknes spektra analizators.

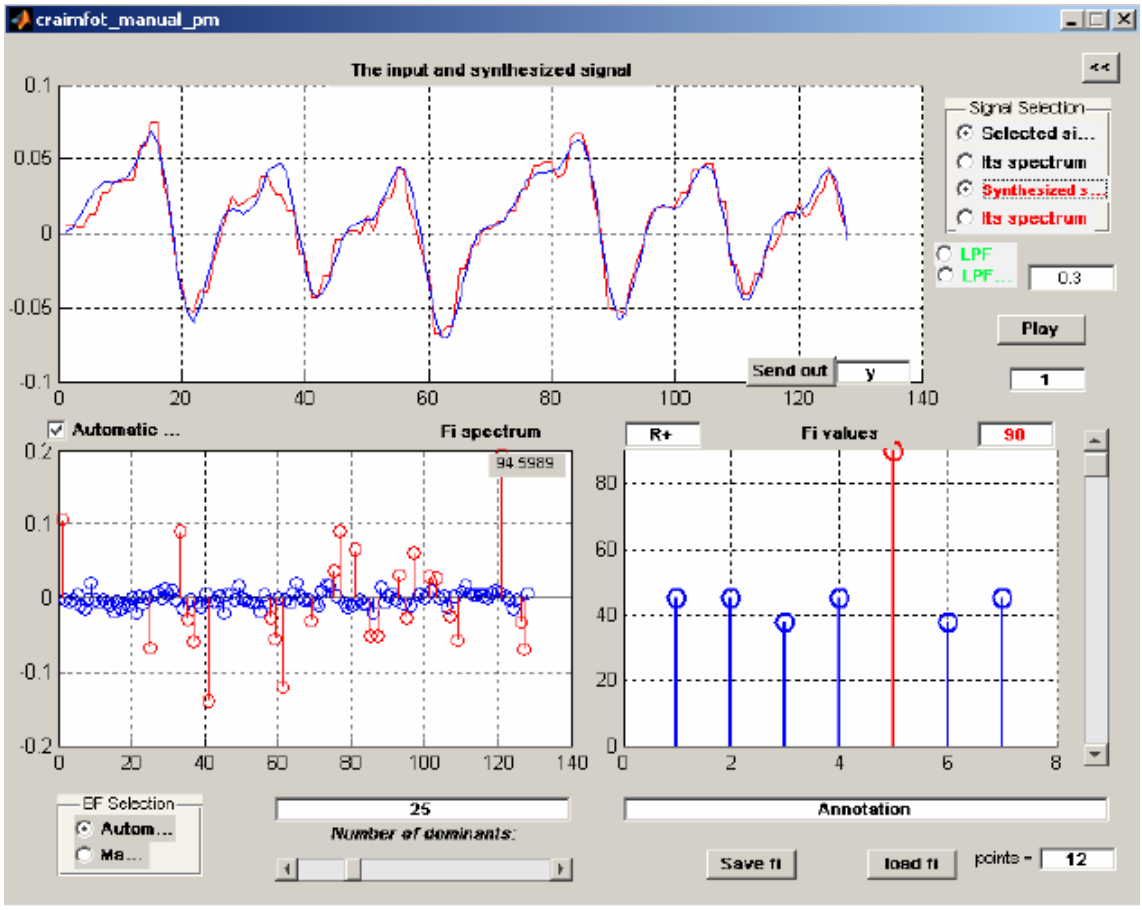
- Uz *RE* balstīto analizatoru iespējams modificēt un lietot arī kā **RABOT** (un **RABOT** apakšklašu, piemēram, **CRAIMOT**) spektra analizatoru.

- Gan virknes, gan paralēlās arhitektūras ir izmantojamas ne tikai spektra analizatoriem, bet arī sintezatoriem.

- Eksperimentāli tika konstatēts, ka analizatora/sintezatora realizācijas versija 1 (2 *FOR* cilpas) ir visātrākā, tomēr tai nepieciešams vislielākais *RE* un līdz ar to arī *LE* skaits. Ņemot vērā, ka sistēma veic gan spektra iegūšanu, gan signāla atjaunošanu, versijai 1 izmantojamo *LE* skaits, ja $n = 3$, vienāds ar ≈ 16000. Versija 3 ir visekeonomiskākā no izmantojamo resursu viedokļa (≈ 3800 *LE*, ja $n = 3$), bet arī vislēnākā. Versijai 2 (viena *FOR* cilpa), kuru var uzskatīt par kompromisu starp *LE* skaitu un realizācijas sarežģītību, nepieciešami ≈ 6400 *LE*. Palielinoties n, izmantojamo *LE* skaits pieaug strauji, piemēram, versija 1, ja $n = 4$, nav realizējama uz izvēlētā *FPGA* (*EP2C35F672C6*).

4.3. Signālu analizators-sintezators

Autora maģistra darba [14] ietvaros tika izveidots uz leņķiem bāzētu pārveidojumu virtuālais runas signālu analizators-sintezators (*SANSYN*) (skat. 4.7. attēlu), par kuru pirmoreiz ir ziņots [50]. Šis līdzeklis ir iekļauts arī atbilstošajā MatLab līdzekļu kopā [P8]. Bez



4.7. att. Runas signāla kompresijas piemērs

tam *SANSYN* ir ļoti būtisks kā palīglīdzeklis uzsākot eksperimentālo *FPGA* ierīču prototipēšanu, piemēram, veidojot eksperimentālo spektra analizatoru [P2]. Attēlotajā līdzeklī, optimālo parametru (leņķu) piemeklēšana tiek veikta pusautomātiski, pa vienam piemeklējot optimālo leņķi. Pēdējās izstrādātā līdzekļa versijās ir iebūvēta arī nelineārās optimizācijas funkcija, kas ļauj piemeklēt optimālo pārveidojumu automātiski. Attēlotajā piemērā tiek veikta latviešu valodas fonēmas "U" spektra iegūšana uz leņķiem bāzētu pārveidojumu bāzē – 95% signāla enerģijas, piemeklējot optimālos leņķus, tiek sakoncentrēti 20% (25 no 128) *BF*-u. Ar izstrādāto līdzekli iespējams arī attēlot un atskaņot no spektra koeficientiem sintezēto audio signālu. Līdzeklis ļauj darboties ne tikai ar audio, bet ar jebkuriem citiem akustiskajiem signāliem t.sk. arī ar biomedicīniskajiem signāliem [51]. Ar līdzekli iegūtās laika diagrammas un spektri ir redzami publikācijās [P1], [P2], [P8] un [3].

Rezumējums

Iegūtie rezultāti

- Ir izveidots virtuālais rīks, kurš ir iekļauts atbilstošajā *MatLab* rīkkopā [P8] (kā **phiansyn**) un ir paredzēts akustisko signālu analīzei un sintēzei.

Secinājumi

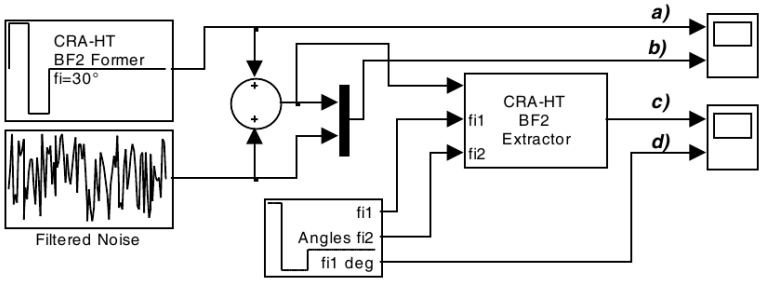
- Izveidotā līdzekļa iestrādes izmantotas fī-pārveidojumu MatLab rīkkopas veidošanā [P8] un līdzeklis ir ļoti nozīmīgs, veidojot eksperimentālās *FPGA* ierīces, jo ļauj iebūvēt dažādus pārveidojumus, iegūt spektrus un sintezētus signālus.
- Pēdējās versijās iebūvētā funkcija ļauj piemeklēt optimālo fī-pārveidojumu izvēlētajiem signāliem un paver jaunas iespējas signālu kompresijas jomā.
- Piemeklējot optimālus leņķus, ar salīdzinoši nelielu *BF* skaitu ($\approx 20\%$) iespējams iegūt kvalitatīvu runas signālu (subjektīvi novērtējot) [P1], [P2], [P8], ko apstiprina arī eksperimenti ar *FPGA* sintezatoru (materiāls vēl nav publicēts),
- Līdzekļa modularitāte un atvērtā struktūra padara to par ērtu palīglīdzekli *FPGA* ierīču prototipēšanai arī nākotnē.

4.4. Hārveidīgie filtri

Publikācijās [P3], [P6], [P8] ir aplūkoti jauna veida ortogonālie filtri, kas pēc savas struktūras ir līdzīgi ortogonālajiem veivletu dekompozīcijas-rekonstrukcijas filtriem. Galvenā atšķirība ir tā, ka filtri ir pārskaņojami ar parametriem (rotācijas leņķiem). Ja klasiskajos veivletu filtros tiek izmantots tas vai cits konkrēts ortogonālais veivletu pārveidojums, tad jaunā veida filtros ir izmantojams praktiski bezgalīgs klāsts pārveidojumu, kuri ir maināmi. Filtrus var darbināt selekcijas/režekcijas, spektra analizatora/sintezatora un citos režīmos. Iespējams, ka [P6], [P8] pirmoreiz ir aprakstīts signāla formas rezonanses efekts (*Signal Shape Resonance*) (tas gan neietilpst promocijas darba aizstāvamo tēžu lokā). Ja filtrs ir noskaņots uz izvēlēto signāla formu un šis signāls ir ortogonāls traucējošajam signālam, tad ir iespējama izvēlētā signāla ideāla izdalīšana (selekcija), ja vien tā amplitūda ir lielāka par noapaļošanas kļūdu. Nederīgā signāla nospiešanas (režekcijas) un ortogonalitātes gadījumā nospiežamā signāla amplitūda var būt pēc patikas liela, bet, protams, tai ir jābūt mazākai par piesātinājuma līmeni. Filtrācijas tīrība ir atkarīga no atdalāmo signālu ortogonalitātes pakāpes [P8]. Minētajās publikācijās [P3], [P6], [P8] ir aplūkoti trīs dažādi ilustratīvi piemēri:

- impulsveida signāla izdalīšana no aditīva trokšņa,
- impulsveida signāla izdalīšana no maskējošas impulsu virknes,
- sinusoīdai piesummēta impulsveida signāla nospiešana.

Šajā piemērā (4.8. att.) parādīts kā no aditīva trokšņa tiek atdalīts impulsveida signāls, izmantojot ortogonālus, uz rotācijas leņķiem bāzētus filtrus. Ja impulsveida signāls ir ortogonāls troksnim (praksē tas ir maz varbūtīgs gadījums), tad piemeklējot pārveidojuma parametrus tā, ka signāls sakrīt ar kādu no ortogonālā pārveidojuma *BF*-ām, signālu

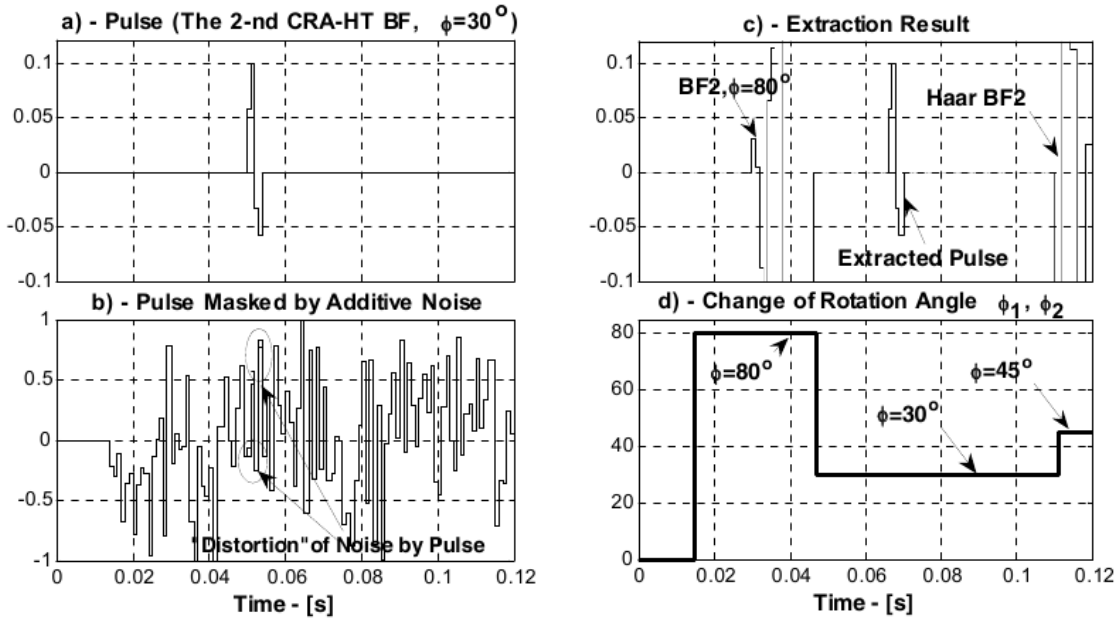


4.8. att. Signāla filtrācijas no trokšņa testa shēma (*SIMULINK*) [P3]

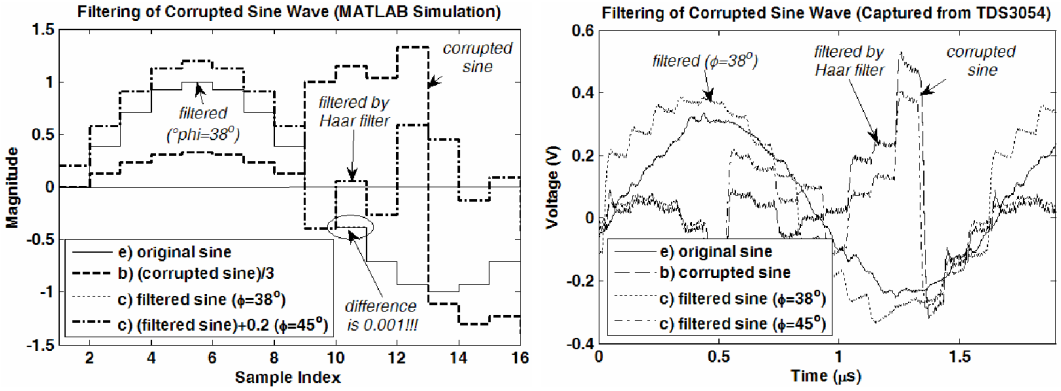
varam ideāli atdalīt no trokšņa, pat ja tā amplitūda ir krietni mazāka par trokšņa amplitūdu, bet lielāka par noapaļošanas kļūdu. Šajā demonstrācijā mākslīgi tika nodrošināts, ka pievienotais impulsveida signāls ir ortogonāls troksnim.

4.9. attēlā parādītas signālu filtrācijas laika diagrammas. Savukārt kropļotas sinusoīdas attīrīšanu (nospiešanu) no impulsveida signāla, kura amplitūda daudzkārt pārsniedz sinusoīdas amplitūdu demonstrē 4.10. attēls. Šajā piemērā savstarpējā ortogonalitāte netiek pilnīgi nodrošināta, taču filtrācijas efekts tāpat ir satriecošs. Ja kropļotajai sinusoīdai harmonisko kropļojumu koeficients ir 100% (tā ir piemeklēta traucējuma amplitūda), tad pēc filtrācijas tas samazinās līdz 0.0004 %! Turpretī, izmantojot Hāra veivletus iegūstam 12.5 %, kas ir tūkstošiem reižu sliktāk. Attēlā ir redzami simulācijas dati un tieši no *FPGA* aparatūriski iegūtās oscilogrammas.

Izveidoto filtru var lietot kā efektīvu reāla laika impulsu izdalīšanas/nospiešanas līdzekli. Filtru var piemērot arī ne-impulsveidīgiem signāliem un filtrācijas kvalitāte tieši tāpat kā impulsveidīgajiem signāliem ir atkarīga no derīgā signāla un traucējošā signāla



4.9. att. Vienas CRA-HT bāzes funkcijas filtrācija no trokšņa [P3]



(a) *MatLab* simulācija (b) *FPGA* realizācija

4.10. att. Ar impulsu kropļota sinusa filtrācijas piemērs [P6]

savstarpējās ortogonalitātes pakāpes. 19. tabulā apkopti daži *DeRe* filtra parametri (divām dažādām realizāciju versijām [P6]), kā arī izmantojamo *FPGA* resursu novērtējums. Filtrējamo impulsu formu nosaka filtra parametri (leņķi). Modificējot attēloto **CRA-HT** filtru iespējams iegūt **CRAIM-HT** filtru (skat. 1.2.3. nodaļu), kuram vadības parametru skaits ir lielāks un līdz ar to arī lielāka ir filtrējamo impulsu formu daudzveidība. Mūsuprāt, **CRAIM-HT** gadījumā ir vislabākā parametru skaita un impulsu formu dažādību attiecība.

19. tabula. DeRe filtra parametri [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	$1400 \cdot n + 15 \cdot (2^{n+1} - 2)$

Rezumējums par publikācijām [P3], [P6] un [P8]

Iegūtie rezultāti

Ir iegūti sekojoši rezultāti:

- Praktiski ir izveidoti, uz *FPGA* balstīti eksperimentālie ortogonālie parametriskie **RA-HT** filtri.
- Apkopoti divu parametrisko filtru arhitektūru *FPGA* realizācijas parametri.

Secinājumi

- Analizējot literatūru ir konstatēts, ka izveidotie parametriskie filtri ir oriģinālas izstrādnes,
- Izveidoto filtru dekompozīcijas daļa ir izmantojama arī kā spektra analizators, bet rekonstrukcijas daļa kā signālu sintezators atbilstošo pārveidojumu bāzē,
- Izveidotie filtri ir līdzīgi klasiskajiem veivletu filtriem, bet atšķiras ar to, ka ortogonālo veivletu filtros viens vai otrs konkrētais ortogonālais pārveidojums ir nofiksēts, bet izveidotajos filtros tas ir maināms,
- Kā rāda eksperimenti ar izveidotajiem **CRA-HT** un **CRAIM-HT** filtriem, tie ir efektīvi dažādu formu impulsveida signālu izdalīšanā/nospiešanā un filtrācijas efektivitāte ir atkarīga no filtrējamā un traucējošā signāla savstarpējās ortogonalitātes pakāpes. Tā, piemēram, ar impulsveida signālu kropļotai sinusoīdai, harmonisko kropļojumu koeficientu izdevās samazināt vairāk nekā 200000 reizes, kas ir vairāk nekā 30000 reižu labāk nekā ar veivletu filtru.
- Modificējot izveidotos filtrus, tos ir iespējams izmantot ne tikai inpulsveida signālu, bet arī signālu ar laikā (telpā) izkliedētu enerģiju (piemēram, ar **CRAIMOT** *BF* formu) filtrācijai un filtrācijas efektivitāte būs atkarīga no atdalāmo signālu ortogonalitātes pakāpes.
- Konstatēts, ka filtrus realizējot *FPGA* ar izvērsto (enrolled) *CORDIC* algoritmu ([P4]-[P7], [P9]), vienas nolases apstrādes laiks praktiski nav atkarīgs no takts frekvences un, piemēram, *EPC35F672C6* čipam tas ir aptuveni 70 (40) ns (9 iterācijām).
- Apkopoti divu parametrisko filtru arhitektūru *FPGA* realizācijas parametri (atkarībā no *RE* elementu skaita) un konstatēts, ka minimālu *FPGA* resursu patēriņu, ja $T_s = 70$ns iegūst, izmantojot otrā veida arhitektūru (*version 2*) [P6].
- Ir atrastas (eksperimentāli) aptuvenas izteiksmes loģisko elementu patēriņa novērtējumam (sk. 19. tabulu). Tā, piemēram, pie filtra pakāpju skaita $n = 3$, *ver. 1* - ≈ 19600, *ver. 2* – ≈ 4320,

- Ir konstatēts, ka no vienas puses filtra versija 1 ir vienkāršāk izveidojama, bet loģisko elementu patēriņš ir tik liels, ka konkrētajam *EP2C35F672C6* čipam iespējamais realizējamo filtra pakāpju skaits ir trīs reizes mazāks nekā versijai 2, bet ātrdarbības ir salīdzināmas.

- Vairāku *VHDL FOR* ciklu izmantošana (*ver. 1*) atvieglo filtra izveidi, jo tādā veidā realizētam filtram, rezultāts tiek iegūts uz katru takts impulsu (*VHDL FOR* cikli filtru izvērš kā paralēlu *FPGA* struktūru), bet *FOR* ciklu skaita samazināšana, sarežģī filtra vadību, bet samazina izmantojamo *LE* skaitu (*RE* darbība jāizvērš laikā).

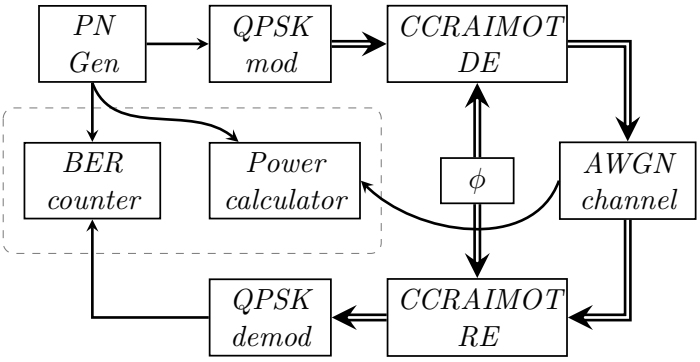
- Izmantojot *CORDIC* algoritmu, filtra realizācijai nav nepieciešami reizinātāji (*DSP* elementi), tāpēc šādus filtrus var realizēt arī zemākas klases *FPGA*, piemēram, *Cyclone I*, zaudējot ātrdarbību.

4.5. Nesinusoidālas frekvenčdales datu pārraides sistēmas prototips-simulators

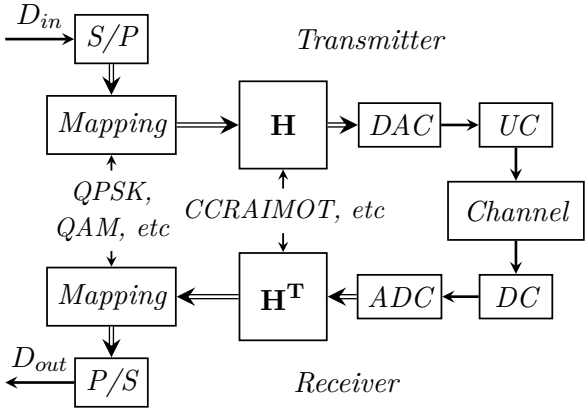
Dažādu pārraides kanāla traucējošo īpašību (galvenokārt selektīva frekvenču vājinājuma) dēļ, datu pārraidē plaši tiek pielietots sinusoidālas frekvenčdales (*OFDM*) princips– datu bloka biti[1] (datu bloka lielums atkarīgs no konkrētā *OFDM*) tiek modulēti ar sinusoidālu signālu, kas ir ortogonāls visiem pārējiem. Šo modulāciju veic ar *IFFT*. Modulācijas veikšanai var izmantot arī nesinusoidālus signālus, kā, piemēram, dažādus veivletus [47]. Literatūrā šādas sistēmas sauc par *OWDM* (***O**rthogonal* **W**avelet **D**ivision **M**ultiplexing) [48], [49]. Publikācijā [53] pirmoreiz ir veikta klasiskā OFDM salīdzināšana ar uz nesinusoidālām ortogonālām funkcijām (Adamāra) balstītas frekvenčdales sistēmu. Japāņu pētnieks Oka [54] arī runā par nesinusoidālu ortogonālu pārveidojumu izmantošanu datu pārraides sistēmās. Izmantojot parametriskos uz rotācijas leņķiem bāzētos pārveidojumus, datus iespējams modulēt ar nesinusoidāliem signāliem. Tā kā pārveidojumi ir parametriski, tad nesinusoidālo signālu formu iespējams mainīt.

Šādas sistēmas izveide (ieskaitot datu pārraides kanāla izveidi) *FPGA* ļauj veikt ļoti garu bināru datu virkņu pārraides simulāciju, kas citās simulācijas vidēs (piemēram, *Simulink*) prasītu stipri vairāk laika. Sistēmas lielākais trūkums – uz *FPGA* jāizveido ne tikai uz leņķiem bāzētais pārveidojums, bet arī signāla ģenerators (4.11. attēlā *PN Gen*), mērelementi (*BER counter* un *Power calculation*), digitālās modulācijas elementi (*QPSK mod* un *QPSK demod*), pārveidojuma vadība (ϕ) un datu pārraides kanāls (*AWGN channel*). Uz *FPGA* realizētas sistēmas lielākā priekšrocība – simulācija tiek veikta reālā laikā ar 10 MHz *pn* ģeneratora takts frekvenci, kas nozīmē, ka, lai iegūtu *BER* (Bit Error Rate) lielumu ar kārtu 10^{-6}, nepieciešama viena sekunde.

[1] parasti bitus modulē, izmantojot kādu no kvadratūrajām modulācijām (*BPSK*, *QPSK*, *4QAM*, utt.)



4.11. att. Vienkāršota nesinusoidālas frekvenčdales datu pārraides sistēmas ciparu daļas blokshēma [P10]

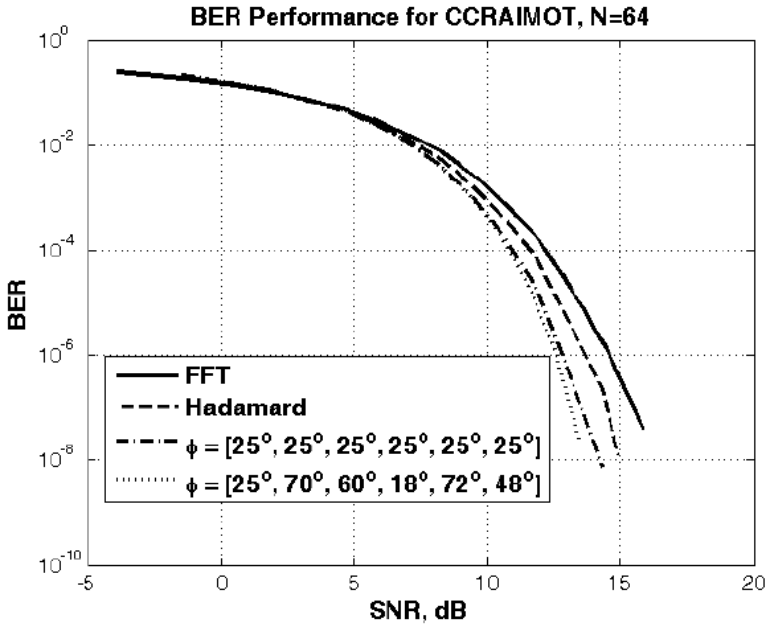


4.12. att. Vienkāršota nesinusoidālas frekvenčdales datu pārraides sistēmas blokshēma [P10]

Simulāciju veikšanai tika izstrādāti divu veidu datu pārraides kanāli – *AWGN* kanāls (kanāls ar aditīvu balto Gausa troksni) un kanāls ar signāla visu frekvenču komponenšu vienmērīgu vājinājumu (*FFC*) un *AWGN*. Summējot vairāku *FFC* iedarbību iegūst sarežģītākus kanālus (*FSFC*), bet tādā gadījumā nepieciešami sarežģītāki kanāla ekvalizācijas algoritmi, bez kuriem nav iedomājams *BER* aprēķins un visa sistēma kopumā. *OFDM*

20. tabula. Datu pārraides sistēmas parametri (*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 %



4.13. att. *BER* līknes pie dažādiem pārveidojumiem ar AWGN kanālu [P10]

kanāla ekvalizācijas un sinhronizācijas algoritmi [52] nedarbojas ortogonālu nesinusoidālu signālu frekvenčdales (*GONDM*) gadījumos [55].

No 4.13. attēla redzams, ka arī ar *AWGN* kanālu *BER* līknes dažādiem pārveidojumiem ir dažādas.

20. tabulā apkopoti izstrādātās datu pārraides sistēmas pamatparametri.

Rezumējums par publikāciju [P10]

Iegūtie rezultāti

- Ir izveidots uz *FPGA* balstīts datu pārraides sistēmas ar vispārināto ortogonālo nesinusoidālo frekvenčdali (*GONDM*) simulators-prototips, kurš ietver sevī
  - uz **CCRAIMOT** funkcijām balstītu modulatoru,
  - baltā trokšņa ar Gausa sadalījumu (*AWGN*) kanāla simulatoru,
  - kombinētā kanāla simulatoru, kurā ir ietverts gan *AWGN* gan fedinga (*FFC*) modelis,
  - pseidogadījuma skaitļu ģeneratoru (pn-ģeneratoru),
  - dažādus palīgmodulatorus (*QPSK*, *QAM* utt.),
  - jaudas kalkulatoru un *BER* skaitītāju.
- Ar izveidotā simulatora-prototipa palīdzību ir iegūtas provizoriskas eksperimentālas *BER* līknes atkarībā no trokšņa līmeņa kanālā.

Secinājumi

- *GONDM* modulatorā/demodulatorā ir ērti izmantojama DEkompozīcijas/REkonstrukcijas filtru arhitektūra, kas minimizē loģisko elementu skaitu salīdzinājumā ar paralēlo arhitektūru.
- *GONDM* ir daudzsološa *OFDM* alternatīva, jo būtiski (praktiski bezgalīgi) papildina frekvenčdalē izmantojamo ortogonālo modulācijas veidu klāstu, kas, savukārt, kā rāda provizoriskie eksperimenti ar **CCRAIMOT** funkcijām, ļauj pieskaņot modulatoru pārraides kanālam un pārraidāmajiem datiem, vismaz par kārtu samazinot *BER* pie attiecības signāls/troksnis lielākas par 12 dB.
- Salīdzinoši liela resursu patēriņa dēļ (simulatorā-prototipā izmantoti visi uz *FPGA* esošie *DSP* elementi (konkrēti *EP3SL1501152C2*), bet reizinātāju veidošana no loģiskajiem elementiem, kā minēts 3.3. nodaļas secinājumos, nav efektīva), aprakstītā *GONDM* 64 kanālu sistēma1 ir realizējama tikai relatīvi augstas klases *FPGA* (*Stratix III* un jaudīgākās).
- *FPGA* realizētā sistēmas simulācijas platforma ļauj īsā laikā ($\sim$ 1 s) iegūt *BER* līknes ar kārtu 10^{-6} un tas ir simtiem reižu (aptuveni 20-40 min.) ātrāk nekā *Simulink*-ā vai desmitiem reižu ātrāk nekā, piemēram, izmantojot *Simulink*-a reālā laika līdzekli (*RTW*).
- Veidojot konkrēto *FPGA* prototipu-simulatoru, ir iegūta vērtīga pieredze, kuru var vispārināt un izmantot arī citās jomās, kurās ir nepieciešami liela apjoma skaitļošanas (simulācijas) darbi (piemēram, materiālu zinātnē). *FPGA* prototips-simulators ir daudzsološa alternatīva paralēlajiem skaitļotājiem, jo ļauj veidot reālā laikā strādājošas kompaktas ierīces ar visdažādāko pielietojumu.
- Pilnas *GONDM* datu pārraides sistēmas izstrādei ir nepieciešami lielāki pētnieku un nopietnāki finanšu resursi. Šeit ir apskatīta tikai daļa no potenciāli nepieciešamajiem pētījumiem. Tuvākajā laikā ir aktuāla (darbi šajos virzienos jau notiek):
  - datu pārraides sistēmas sinhronizācijas izstrāde,
  - kanāla ar selektīvu frekvenču vājinājumu (*FSFC*) simulācija,
  - kanāla ekvalizācijas algoritmu izstrāde,
  - parametru pieskaņošanas algoritmu izstrāde.

Nobeigums

Fī-pārveidojumu tēmas, kas ir tik plaša, ka tās pilnai aptveršanai nepieciešami lieli cilvēku un finanšu resursi, izstrādes īpatnības ir tādas, ka:

- tēmas ietvaros līdzsvaroti un vienlaicīgi tiek pētīta (attīstīta) gan fī-pārveidojumu teorija, gan arī pārveidojumu praktiskās īstenošanas iespējamība,
- tēmā ir iezīmējušies vairāki attīstāmie virzieni: signālu analīze, sintēze, kompresija, filtrācija, vispārinātā frekvenčdale, attēlu apstrāde, atbilstošu praktisku ierīču izveides automatizācija, utt.

Pašreizējā tēmas attīstības stadija ir saistīta ar augsnes un iedīgļu sagatavošanu turpmākajiem potenciālajiem pētījumiem. Izstrādātais promocijas darbs, autoraprāt, ir uzskatāms par pabeigtu, ja to interpretē kā iestrādi un sagatavošanās posmu plašākiem pētījumiem.

Paveiktā darba galvenais akcents ir vērsts uz pārveidojumu praktiskās īstenošanas iespējamības novērtēšanu, kādēļ darbā apskatītas vairāku uz fī-pārveidojumiem balstītu, vairākos virzienos vērstu ierīču realizācijas *FPGA* un sniegti dažādu to parametru salīdzinājumi. Apskatītās ierīces ir pirmās izveidotās fī-pārveidojumu ierīces, kuras var uzskatīt par sākuma platformu reālu produktu izstrādei. Eksperimentālo ierīču izveide ļauj novērtēt fī-pārveidojumu realizācijas sarežģītību un veikt sintēzes algoritmu uzlabojumus, kā arī novērtēt un uzlabot ierīču parametrus (gan samazināt izmantojamo *FPGA* loģisko elementu skaitu, gan paaugstināt to ātrdarbību).

Būtiska iestrāde ir veikta ortogonālo filtru jomā, kas ļauj veidot parametriskus filtrus ar unikālām īpašībām un ir acīmredzams pamats ne vienam vien pētījumam (t.sk. arī promocijas darbiem) dažādās jomās, piemēram, sonāru un radaru signālu apstrādē. No konjunktūras viedokļa, kas, galvenokārt, ir saistīta ar *OFDM* sistēmai alternatīvu datu pārraidi, nozīmīga ir eksperimentālā vispārinātās frekvenčdales sistēmas prototipa-simulatora izveide. Tā ir sākuma platforma jaunu frekvenčdales sistēmu izstrādei un reizē simulācijas platformas alternatīva virtuālajiem simulatoriem. Vispārinātās frekvenčdales modulācijā veiktās iestrādes ir potenciāli izmantojama ne tikai datu pārraidē un ciparu apraidē, bet arī jau pieminētajos radaros un sonāros. Minētās modulācijas veida izpēte ir novedusi pie pastiprināta vispārinātā pārveidojumu pamatelementa (*EGURM*) izpētes.

EGURM pieraksta forma ļāva salīdzinoši vienkāršā veidā aprakstīt visas iespējamās Jakobi matricas struktūras, kas noveda pie automatizētas *RE* sintēzes sistēmas izveides. Šajā sistēmā, apvienojot gan teorētisko pusi (*EGURM* struktūru izvēle, matemātisko sakarību pārveidojumi, u.c.), gan praktisko (*VHDL* simulācijas, izmantojamo *FPGA* resursu noteikšana), būtiski mainīta fī-pārveidojumu sintēzes forma – fī-pārveidojumu pamatelementu (*RE*) sintēze tiek veikta automatizēti. Lai sistēmu varētu uzskatīt par pilnīgi automatizētu, vēl ir jāpilnveido signāla nolašu vārda garumu aprēķināšanas algoritms, kurš

jāsavieno ar fiksētā punkta aritmētikas *FPA* kļūdu novērtēšanas rīku. Pašreiz rīks ļauj iegūt kļūdu atkarībā no *RE* signālu izvēlētā nolašu vārda garuma, bet vārda garumi tiek koriģēti manuāli, kādēļ ir jāveic tādi uzlabojumi, kas ļautu automātiski no uzdotās kļūdas atrast *RE* signālu nolašu vārda garumus.

Kā turpmāko darbu var minēt automatizētas fī-pārveidojuma sintēzes sistēmas (*UNITIT*) izveidi, kurā tiktu apvienoti kokveida (arī paralēlie) dekompozīcijas-rekonstrukcijas algoritmi un *EGURIT*. Bez tam tuvākās nākotnes aktualitāte ir *EGURIT* papildināšana ar komplekso *CORDIC* un atbilstošu resursu novērtēšanas algoritmu, kas ļautu ierīcēs rotācijai izmantot gan, jau iebūvēto, klasisko rotācijas algoritmu, gan *CORDIC*, un tādējādi optimizēt loģisko elementu patēriņu un ātrdarbību. Ne mazāk aktuāls tuvākās nākotnes uzdevums ir *EGURIT* un *UNITIT* sistēmu adaptācija arī citām izstrādes platformām (*Xilinx*, *Synopsis* u.c.) kā arī atbilstošo eksperimentālo *ASIC* čipu izstrāde un pie tā īstenošanas tiek strādāts.

Fī-pārveidojumu tēma ir atvērta intensīviem turpmākajiem pētījumiem. Tā, piemēram, ir aizsācies darbs pie 2-D signālu fī-pārveidojumu *FPGA* realizāciju izstrādes. Pirmajā publikācijā, kas veltīta šai tēmai (visu autora publikāciju sarakstā, Nr. 11), apskatīti ar atmiņas vadību saistītie jautājumi un 2-D pārveidojumu, kas balstīti uz *RE*, realizācijas arhitektūras.

Literatūras saraksts

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

[2] RTU, "RTU Doktorantūra 2008./2009." RTU izdevniecība, Rīga-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.