Scientific Journal of Riga Technical University
Telecommunications and Electronics
2011
Volume 11

Automation of FPGA Implementation of Unitary Transforms Based on Elementary Generalized Unitary Rotation

Gatis Valters[1], Peteris Misans[2], [1-2]*Riga Technical University*

Abstract **- This paper presents the basics of complex Jacobi-like Elementary Generalized Unitary rotation (EGU-rotation) and the basics of building of parametrical unitary transforms using EGU-rotations. EGU-rotation matrix (EGURM) is multifaceted and takes 64 different shapes and up to several hundred faces for different sets of used angles. For EGURM addressing, up to 8 parameters are necessary. Multiplier-adder based rotation algorithm is discussed in more detail because of its suitability for high speed applications. The number of multipliers and adders varies in a wide range. The development of devices based on unitary transforms is a task of high complexity. The present work can be treated as a basis for or a road map to development of automated tools.**

Keywords**: Unitary Transforms, Parametrical Transforms, Jacobi Rotation, FPGA, Fixed Point Arithmetic, Digital Signal Processing**

I. INTRODUCTION

This paper deals with the necessity and problems of automation to implement unitary transforms based on unitary Jacobi-like rotations (elementary generalized unitary (EGU) rotations). Jacobi rotator is the basic building block for modern signal processing algorithms and devices. It has been widely used in the FPGA implementation of complex SVD, EVD, eigenvalue computation and their applications, for example, [1]-[7]. Kim et al [8] use FPGA CORDIC-based Jacobi processor for the estimation of direction of arrival (DOA) in cellular wireless base station applications. It is also typical that rotation has been used for beamformers, e.g. [9], [10]. Researchers use different solutions for the rotator arithmetic. For example, in [11] distributed arithmetic has been used to re-arrange a complex rotation into a form more suited to FPGA implementation. It seems that Otte and his colleagues are pioneers in the factorization of complex Jacobi rotation by several real rotations [10]. In rotation implementation, CORDIC-algorithm dominates and such approach is motivated mainly by the relative low hardware and power consumption. However, FPGA-based CORDIC solution may be not the optimal one from a speed point of view [12] even for parallel (enrolled) CORDIC [13] if FPGA is equipped with fast built-in multipliers.

Recently our team has been working on different FPGA (also ASIC) implementations and applications of generalized unitary transforms, e.g. [14]. The automation of the implementation arises as a task of primary importance because of design complexity. The present work concerns the first in the range of steps to deal with development of our tools for the automation of implementation process focused on unitary transforms. The second step is highlighted in [15] where we describe a tool for the automation of VHDL code generation targeted on elementary generalized unitary transform. The current paper discusses mainly "what and why", but [15] - "how and what is done". It is well known that in the recent decade, Simulink and its blocksets - Xilinx System Generator and Altera DSP Builder have been used widely for automation of VHDL code generation and prototyping of FPGA devices. In IEEE MDL [16] we can find close to hundred publications about the use of both tools, but if we try to look for the investigations narrowly targeted on the topic given in the title of this paper, we see the lack of papers.

A. Terminology

The terms "**E**lementary **G**eneralized **U**nitary **R**otation **M**atrix" (EGURM) and "EGURM-rotator" have been introduced and briefly described in [12]. These terms should be treated as a trial of generalization of well-known terms "unitary Jacobi rotation matrix" and "Jacobi rotator" available in open literature. Further we suppose that EGURM is a unitary four-element and three-parameter (angles) matrix (see below). We also suggest that the term "EGU-rotator" is more preferable instead of "EGURM-rotator".

We introduce here 3 basic terms to describe EGURM and the corresponding rotator:

- *shape* – this means the mutual location of *sin/cos*, the signs before *sin/cos* and ψ/γ in the matrix,
- *face* – this means the chosen shape of matrix with defined values (also variable) of angles ϕ,γ and ψ,
- *EGU-rotator unit* – this means a physical unit that performs EGU-rotation for the given EGURM face. It can be updated after rotation to use another face or replaced by another unit (reconfiguration).

We suggest that the introduced terms serve as "local necessity" and do not make any claim on the "historical originality".

II. BASICS OF EGU-ROTATION

The elementary unitary rotation in the Hilbert space can be presented as a simple product:

$$\mathbf{y} = \mathbf{T} \cdot \mathbf{x}, \quad \mathbf{x} = \mathbf{T}^{T} \cdot \mathbf{y}, \tag{1}$$

where $(\ldots)^{T}$ means the Hermitian transposition of matrix. $\mathbf{x}$ and $\mathbf{y}$ – two-element input vector (signal) and output vector

(spectrum), respectively:

$$\mathbf{x}=\begin{bmatrix} x_{\mathrm{Re}\,k}+j\cdot x_{\mathrm{Im}\,k} \\ x_{\mathrm{Re}\,m}+j\cdot x_{\mathrm{Im}\,m}\end{bmatrix},\quad \mathbf{y}=\begin{bmatrix} y_{\mathrm{Re}\,k}+j\cdot y_{\mathrm{Im}\,k} \\ y_{\mathrm{Re}\,m}+j\cdot y_{\mathrm{Im}\,m}\end{bmatrix}. \tag{2}$$

Lower case indices point to the signal/spectrum sample and to the real/imaginary part of the sample. The matrix **T** represents a four-element unitary rotation matrix that can be defined in many ways (see (3)).

A. Shapes of Rotation Matrix

In the case of real EGURM (γ=0, ψ=*0* in (3)) exist 16 alternatives how to build up a **T** [17], if angle $\phi\in[0, 45°]$, however for unitary (complex valued) rotation we have **64** alternatives of elementary rotation matrices. The next formula allows obtaining the diversity of numerous shapes of unitary rotation matrices for ϕ, γ, $\psi\in[0, 45°]$:

$$\mathbf{T}(\phi,\psi,\gamma,a,b,u)=\mathbf{T}(\phi,\psi,\gamma,abu)=$$
$$\begin{bmatrix} s_{11}(a)sc(b)\cdot e^{sp(u)\,j\cdot\psi} & s_{12}(a)cs(b)\cdot e^{-sg(u)\,j\cdot\gamma} \\ s_{21}(a)cs(b)\cdot e^{sg(u)\,j\cdot\gamma} & s_{22}(a)sc(b)\cdot e^{-sp(u)\,j\cdot\psi}\end{bmatrix} \tag{3}$$

where

$$s_{1,1}=\begin{bmatrix}-&-\\-&-\\+&+\\+&+\end{bmatrix}, s_{1,2}=\begin{bmatrix}-&-\\+&+\\-&-\\+&+\end{bmatrix}, s_{2,1}=\begin{bmatrix}-&+\\-&+\\-&+\\-&+\end{bmatrix},$$
$$s_{2,2}=\begin{bmatrix}+&-\\-&+\\-&+\\+&-\end{bmatrix}, sc=\begin{bmatrix}\sin\phi\\ \cos\phi\end{bmatrix}, cs=\begin{bmatrix}\cos\phi\\ \sin\phi\end{bmatrix},$$
$$sp=\begin{bmatrix}-&+\\-&+\end{bmatrix}, sg=\begin{bmatrix}+&-\\-&+\end{bmatrix}$$

are sign and *sin*/*cos* matrices, but $a\in[1,8]$, $b\in[1,2]$, $u\in[1,4]$ – indices for addressing of elements of matrices (see details in the subsection about addressing).

B. EGURM Based Unitary Transforms

Recently our team has been working on the applications of EGU-rotation in the novel modulation schemes [14]. In this case and also in other applications have been used unitary transforms. The direct/inverse fast unitary transform in the matrix form can be represented as:

$$\mathbf{Y}=(\prod_{p=n}^{1}\mathbf{P}_p\cdot\mathbf{B}(\varphi_p))\cdot\mathbf{X},$$
$$\mathbf{X}=(\prod_{p=1}^{n}\mathbf{P}_p^T\cdot\mathbf{B}^T(\varphi_p))\cdot\mathbf{Y},\quad n\le\log_2(N), \tag{4}$$

where $(\dots)^{\mathrm{T}}$ means the Hermitian transposition of matrix. **X**/**Y** – input signal/spectrum vector, $\mathbf{B}(\varphi_p)$ (further $\mathbf{B}_p$) - sparse unitary matrix presented by (5) and it is named as the **B**lock-wise **U**nitary **G**eneralized **R**otation **M**atrix (BUGRM):

$$\mathbf{B}_p=\begin{bmatrix}\mathbf{T}_{1\,p} & \vdots & 0 & 0\\ & & 0 & 0\\ \cdots & \ddots & \cdots & \\ 0\;\;0 & \vdots & \mathbf{T}_{\frac{N}{2}\,p} & \\ 0\;\;0 & & & \end{bmatrix}, \tag{5}$$

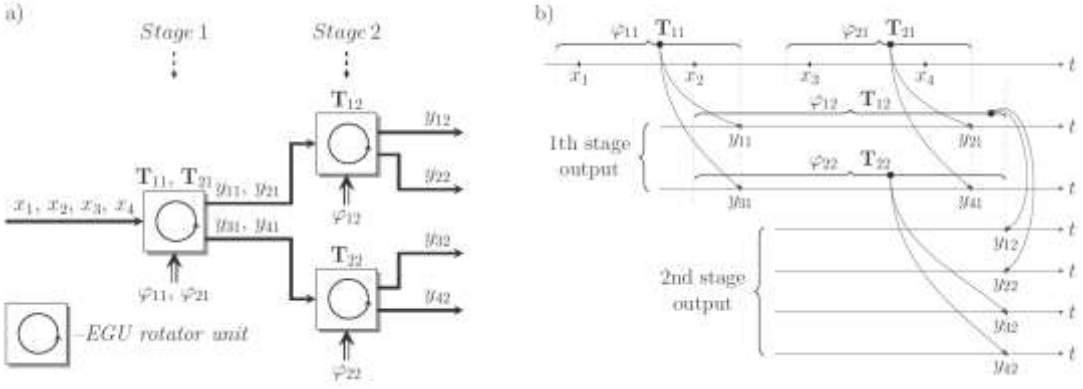
where for the EGURM shape given, for example, by (index set {8,1,4} – see below)

$$\mathbf{T}_{r\,p}(\phi,\psi,\gamma,814)=\begin{bmatrix}\sin\phi_{r,p}e^{j\gamma_{r,p}} & \cos\phi_{r,p}e^{-j\psi_{r,p}}\\ \cos\phi_{r,p}e^{j\psi_{r,p}} & -\sin\phi_{r,p}e^{-j\gamma_{r,p}}\end{bmatrix} \tag{6}$$

and $r\in[1, N/2]$, $p\in[1, n]$. $\mathbf{P}_p$ is a permutation matrix, but N is the size of matrices $\mathbf{B}_p$, $\mathbf{P}_p$ and the length of vectors **X**, **Y**. We can create an infinite number of unitary transforms using BUGRM and the array (matrix with size $N/2\times n\times 3$) of rotation angles [18]. In formula (7) φ_p represents the p-th column of 3-D angle matrix $\mathbf{\Phi}$.

$$\mathbf{\Phi}=\left|\varphi_1,\quad \dots\quad ,\varphi_p,\quad \dots\quad ,\varphi_n\right|,$$
$$where\ \varphi_p=\begin{bmatrix}\phi_{1\,p},\phi_{2\,p},\dots,\phi_{N/2\,p}\\ \psi_{1\,p},\psi_{2\,p},\dots,\psi_{N/2\,p}\\ \gamma_{1\,p},\gamma_{2\,p},\dots,\gamma_{N/2\,p}\end{bmatrix}. \tag{7}$$

When all the rotation structures in (5) are identical, the angle matrix (7) can be simplified, and we obtain parametrical **C**omplex **C**onstant **R**otation **A**ngle **I**n **M**atrix **O**rthogonal **T**ransform (CCRAIMOT) [14]. The fast parallel architecture for implementation of unitary transform described by (4) is possible, but the tree-like structure of algorithm is more useful, because of minimized consumption of hardware resources [12]. Such a structure of unitary transform (also orthogonal filter) with serial data flow is well known from the theory and applications of packet wavelets, but in our case several differences highlighted below appear. A common feature is that we have the same result on the output of tree-like structure and the output of (4) after incoming of N sample block. We demonstrate here a simple example for a better understanding.



a) *Example 1*. We can rewrite the (4) for N=4 and the chosen identical permutation matrices, as:

$$\mathbf{Y} = \mathbf{Y}_2 = \mathbf{P}_2 \cdot \mathbf{B}_2 \cdot \mathbf{Y}_1$$

$$\begin{bmatrix} y_{1,2} \\ y_{2,2} \\ y_{3,2} \\ y_{4,2} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} \mathbf{T}_{1,2} & & 0 & 0 \\ & & 0 & 0 \\ 0 & 0 & & \\ 0 & 0 & & \mathbf{T}_{2,2} \end{bmatrix} \cdot \begin{bmatrix} y_{1,1} \\ y_{2,1} \\ y_{3,1} \\ y_{4,1} \end{bmatrix}, \tag{8}$$

where

$$\mathbf{Y}_1 = \mathbf{P}_1 \cdot \mathbf{B}_1 \cdot \mathbf{X}_1$$

$$\begin{bmatrix} y_{1,1} \\ y_{2,1} \\ y_{3,1} \\ y_{4,1} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} \mathbf{T}_{1,1} & & 0 & 0 \\ & & 0 & 0 \\ 0 & 0 & & \\ 0 & 0 & & \mathbf{T}_{2,1} \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix}. \tag{9}$$

In the corresponding tree-like block diagram (Fig. 1), the following features can be observed: we operate with parametrical EGU-rotator (possible to change the $\varphi_{r\ p}$ in time, if necessary);

- rotator unit should change the face over time (for the first pair of samples $\mathbf{T}_{1,1}$, for the second $\mathbf{T}_{2,1}$ and so on) – this means the cyclic update of the unit parameters or change of the unit over time;
- the number of rotator unit faces decreases, but the unit operating time grows twice with the increase of stage index.

For the reconstruction structure (fan-in), $\mathbf{T}^{\mathrm{T}}$ instead of $\mathbf{T}$ has been used. The above-listed features are useful also for the reconstruction structure in case of reverse indexing (see the second formula in (4)) only. This means that the EGU-rotator at the end (output) of structure operates with matrices $\mathbf{T}^{\mathrm{T}}{}_{1,1}$ and $\mathbf{T}^{\mathrm{T}}{}_{2,1}$ (for N=4). In some special cases, for example, when CCRAIMOT has been used, $\mathbf{T}_{1,1}=\mathbf{T}_{2,1}$, $\mathbf{T}_{1,2}=\mathbf{T}_{2,2}$ and subsequently it is not necessary to change the EGURM face during operation of rotator unit. The listed features are also true for $N=2^n>4$. The main difference between transforms (4) and wavelets is a "chameleon nature" of (4). The known orthogonal (unitary) wavelet transforms are only special cases of (4) [15]. We save a more detailed comparison for the future.

C. The Number of Rotators

The implementation of fast parallel architecture needs $n\times(N/2)$ rotators, but the tree-like structure consumes only $N-1$ rotators. However, that is only correct for the above-mentioned CCRAIMOT-like transforms (one angle per BUGRM matrix). Generally, we should make the update of face or reconfiguration of rotator unit during decomposition (also reconstruction) and finally operate with the number of rotator faces:

$$N-1 \le n_{rot} \le n \times (N/2). \tag{10}$$

D. Addressing of EGURM Shapes and Faces

There are many ways on how to address EGURM shapes. It depends on the initial definition of matrices in (3). Possibly there is a more justified ordering of the shapes (in the sense of initial definition of sign matrices), but we suppose that such a discussion at this moment would be senseless. Further, we will use (3) as a base for addressing of the shape. To build up the necessary rotation matrix shape using (3), first of all, we should select the indices *a*, *b* and *u*. In the second step follows the choosing of corresponding signs and *sin* or *cos* follows. For the addressing of matrix elements in (3) we have chosen the MATLAB-like indexing, where for a matrix interpreted as

TABLE 1.
THE NUMBER OF ELEMENTARY ROTATOR UNITS FOR THE PARALLEL ((4)) AND TREE-LIKE STRUCTURE OF TRANSFORM

N	2	4	8	16	32	64	128
Parallel *(n×(N/2))*	1	4	12	32	80	192	448
Tree like *(N-1)*	1	3	7	15	31	63	127

TABLE 2.
SOME RANDOM EXAMPLES OF ROTATION MATRIX SHAPES FOR DIFFERENT INDEX SETS

ID id1	*Index set {a,b,u}, id3*	*Rotation matrix*
1	{1,1,1} *111*	$\begin{bmatrix} -\sin\phi\cdot e^{+j\psi} & -\cos\phi\cdot e^{+j\gamma} \\ -\cos\phi\cdot e^{-j\gamma} & +\sin\phi\cdot e^{-j\psi} \end{bmatrix}$
6	{1,2,2} *122*	$\begin{bmatrix} -\cos\phi\cdot e^{-j\psi} & -\sin\phi\cdot e^{-j\gamma} \\ -\sin\phi\cdot e^{+j\gamma} & +\cos\phi\cdot e^{+j\psi} \end{bmatrix}$
64	{8,2,4} *824*	$\begin{bmatrix} +\cos\phi\cdot e^{+j\psi} & +\sin\phi\cdot e^{-j\gamma} \\ +\sin\phi\cdot e^{+j\gamma} & -\cos\phi\cdot e^{-j\psi} \end{bmatrix}$

a single column can be used also a single index. This means that the change of sets looks like: {1,1,1}, {1,1,2}, {1,1,3}, {1,1,4}, {1,2,1}, {1,2,2}, …

The table shows some examples of EGURM shapes, their identification numbers (*ID* - named also as *id1*) and corresponding index sets. From the practical point of view more useful is 3-digit index *id3*, for example, 424 instead of {4,2,4}. The shortest way for the mutual conversion of *id1* and *id3* is a simple table, but also the algorithms for extraction of indices *a*, *b* and *u* from *id3* are easy:

$$a = fix(id3/100),\ b = round(fix(id3/10) - 10\times a),\\ u = round(10\times(id3/10 - fix(id3/10))), \tag{11}$$

where *fix()* and *round()* - well-known rounding functions.
We see that correct adressing of EGURM in matrix **B** needs two additional indices *r* and *p* ((5)), where *r* points at EGURM in BUGRM, but *p* at BUGRM. The meaning of *r* index is more complicated in case of tree-like architecture. This index points at the face of EGURM used in the rotator unit:

$$r\in[(q-1)\frac{N}{2^p}+1,\quad q\frac{N}{2^p}],\ q\in[1,2^{p-1}],\ \ p\in[1,\ n], \tag{12}$$

where *q* points at the chosen unit, but *p* points at the *p*-th stage of decomposition filter. An example of indexing for the tree-like structure (*N*=8) is depicted in Fig. 2. The most left nodes in the tree correspond to the EGU-rotator unit at the top of stage in Fig. 1 a). The formula (12) is also useful for the reconstruction structure if we suppose that the first stage is located at the end (output) and the reverse indexing (see above) has been used. Finally, we see that for the correct choice of each of rotation matrices up to 8 (7 – for parallel structure) parameters are necessary – three angles, shape *id*, 3 (2) location indices and the size of transform matrix *N*.

b) Example 2. To operate with the rotation matrix T_{42} (Fig. 2) with shape id3 = 323 (id1=23) that is located at the second stage (p=2) and in the second rotator unit (q=2) we should use such a face of the EGURM:

$$\mathbf{T}(\phi,\psi,\gamma,323,4,2,2,8) = \\ =\begin{bmatrix} \cos\phi_{4,2}e^{-j\psi_{4,2}} & -\sin\phi_{4,2}e^{j\gamma_{4,2}} \\ -\sin\phi_{4,2}e^{-j\gamma_{4,2}} & -\cos\phi_{4,2}e^{j\psi_{4,2}} \end{bmatrix}. \tag{13}$$

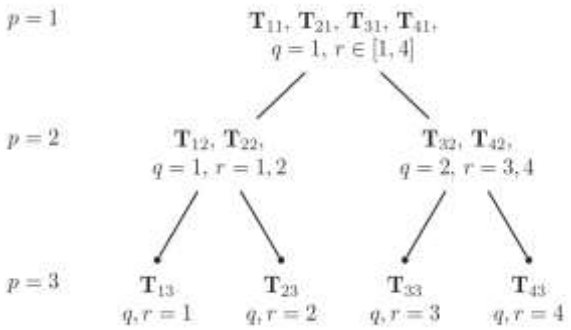


Fig. 2. Decomposition tree structure and addressing example for N=8

We skip here the details about permutation (it follows from (8) and (9)) of tree branches because of relative simplicity of this operation.

III. ROTATION ALGORITHM

A. CORDIC or Classical Algorithm?

Here a "classical algorithm" means based on multiplication(s) (M) and addition(s) (A) (MA-based algorithm). Although in rotation implementation dominates the CORDIC, our conclusion is that both approaches should be used depending on application. Since CORDIC-based algorithms are widely described, we focus our efforts on the classical approach (see below). The main reason for that is an operation speed. Our experience shows that MA-based algorithm can be performed approximately twice faster than the CORDIC-based algorithm for the FPGA equipped with the built-in multipliers [12]. On the other hand, as it follows from the timing diagram Fig. 1 b), we have the increasing of processing time in each stage because of decimation. That leads to reduction of requirements for the speed of FPGA parts used in the stages behind the first one. A good compromise for implementation of tree-like structure can be the mixed solution – MA-based rotation for the first two stages, but CORDIC-based for others. We will report about it in our next publications.

B. MA-based Algorithm

After simple algebraic manipulations on (1)-(3), we obtain the basic expressions for calculations of *y* elements:

$$\begin{aligned} y_{\mathrm{Re}\ k} &= x_{\mathrm{Re}\ m}\cdot cc_\gamma + x_{\mathrm{Im}\ m}\cdot cs_\gamma + x_{\mathrm{Re}\ k}\cdot sc_\psi - x_{\mathrm{Im}\ k}\cdot ss_\psi,\\ y_{\mathrm{Re}\ m} &= x_{\mathrm{Re}\ k}\cdot cc_\gamma - x_{\mathrm{Im}\ k}\cdot cs_\gamma - x_{\mathrm{Re}\ m}\cdot sc_\psi - x_{\mathrm{Im}\ m}\cdot ss_\psi,\\ y_{\mathrm{Im}\ k} &= x_{\mathrm{Im}\ m}\cdot cc_\gamma - x_{\mathrm{Re}\ m}\cdot cs_\gamma + x_{\mathrm{Im}\ k}\cdot sc_\psi + x_{\mathrm{Re}\ k}\cdot ss_\psi,\\ y_{\mathrm{Im}\ m} &= x_{\mathrm{Im}\ k}\cdot cc_\gamma + x_{\mathrm{Re}\ k}\cdot cs_\gamma - x_{\mathrm{Im}\ m}\cdot sc_\psi + x_{\mathrm{Re}\ m}\cdot ss_\psi, \end{aligned} \tag{14}$$

where

$$cc_\gamma = \cos(\phi)\cdot\cos(\gamma),\ cs_\gamma = \cos(\phi)\cdot\sin(\gamma),$$
$$ss_\psi = \sin(\phi)\cdot\sin(\psi),\quad sc_\psi = \sin(\phi)\cdot\cos(\psi), \qquad (15)$$
$$s_\phi = \sin(\phi),\quad c_\phi = \cos(\phi)$$

are substitutions in (14). The present formulas are true for the shape given by (6) (*id3*=814) only. For other shapes, the formulas are similar to (14) and (15), but signs and substitutions can differ from the given.

1) Simplification of Algorithm. It is possible, if we use some restriction on rotation angle values ϕ, γ and ψ .

a) Example 3. Angle values $\gamma = \psi = \pi/4$ reduce (6) to:

$$\mathbf{T}(\phi,\frac{\pi}{4},\frac{\pi}{4}) = \begin{bmatrix} \frac{\sqrt{2}}{2}\sin\phi\cdot(1+i) & \cos\phi \\ \cos\phi & -\frac{\sqrt{2}}{2}\sin\phi\cdot(1-i) \end{bmatrix} \qquad (16)$$

but

$$y_{\mathrm{Re}\,k} = x_{\mathrm{Re}\,m}\cdot c_\phi + \frac{\sqrt{2}}{2}(x_{\mathrm{Re}\,k} - x_{\mathrm{Im}\,k})\cdot s_\phi, \qquad (17)$$
$$y_{\mathrm{Re}\,m} = \dots,\ y_{\mathrm{Im}\,k} = \dots,\ y_{\mathrm{Im}\,m} = \dots,$$

where expressions $y_{\mathrm{Re}\,m}$, $y_{\mathrm{Im}\,k}$, $y_{\mathrm{Im}\,m}$, are similar to $y_{\mathrm{Re}\,k}$, and differ by indices and signs only. The total number of simplifications of (14) (like (17)) depends on the number of sets of angles used, and diversity of formulas like (17) increases significantly versus the length of angle set:

$$M_{anglesets} = (l_{set})^3 - 1,\quad for\ l_{set} > 3, \qquad (18)$$

where $M_{angle\ sets}$ – the number of angle sets, l_{set} – set length. For example, in the case of 4 different values for each of angles (e.g. *variable*, *0*, *π/4*, and *π/2*), we have **63** different simplification combinations.

2) Number of Operations. It depends on the set of chosen angles used for evaluation of rotation matrix. This number varies within a wide range. For (14) (including (15)), when we change all (three) angles, we should perform 20 multiplications (M) and 12 additions (A). In case when some angles are constant (e.g. (16)) there is also the multiplication by constant (MC). The next table is true for the shape (6) (*id3 = 814*).

The number of operations depends also on internal architecture of rotator. In case when we save some products (e.g., $cc_\gamma \times x_{Re\ k}$) in memory, we can also avoid some multiplications (correspondingly, $cc_\gamma \times x_{Re\ m}$), if *m=k+1*.

TABLE 3.
THE NUMBER OF OPERATIONS FOR SOME SETS OF ANGLES AND ROTATORS WITHOUT USING MEMORY

ϕ	*var*	*var*	*var*	π/2	*var*	*var*	π/2
ψ	*var*	*var*	π/4	π/4	0	π/2	π/2
γ	*var*	0	π/4	var	0	π/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

IV. ROAD MAP TO DESIGN AUTOMATION

From the previous sections it follows that FPGA implementation of unitary transforms is a task of high complexity. We should manage quite different things from symbolic computation and text processing to HDL coding and testing to reach optimal results from the viewpoint of performance, resource consumption and accuracy. The main design automation points follow below. Some of them are common for almost any FPGA-based DSP design, but others are specific for EGU-rotation based design mainly. The main goal of automation is to shorten the way from "foggiest idea" to hardware. On the other hand, since expressions (4) give an infinite freedom to the creation of transforms and their applications, we should usually remember about restrictions on hardware resources and speed. This is a reason why we need to get a rapid feedback about consumption of resources and timing for our project. We must be sure whether the proposed algorithm is implementable or not. Finally, the testing is necessary. That includes algorithm simulation (e.g. Matlab/Simulink), HDL simulation (using, e.g. ModelSim) and result comparison.

C. Unitary Transform Building Management

1) Choice of structure. Presently we have two alternatives for implementation of transform – parallel and tree-like ones. If we want to work with a serial flow of samples and run a device as an orthogonal filter, the tree-like structure is preferable, but in case when we need manipulations with spectrum only, the parallel structure is more acceptable.

2) Necessity for Symbolic Computation. EGURM, to a great extent, is multifaceted for the manual and reliable management. In other words, the number of EGURM shapes is too great (up to 64). The same problem arises with a big diversity of faces (up to several hundred). We should use symbolic computation to get the correct expression for the chosen shape and finally also for the face evaluated by the selected set of angles. Symbolic computation is very important also in the case of MA-based algorithms to obtain expressions like (17).

3) Algorithm Type. We should also manage the selection of type of algorithm for each EGU-rotator (complex CORDIC, MA-based or others) in the context of resource consumption and requirements for accuracy and timing.

4) Text Processing. For the MA-based algorithm it is necessary to expand the obtained EGU-rotation expressions (like (14) and (17)) into separate elementary expressions like two-term product and two- (up to four) term sum. Such elementary expressions are useful for further HDL coding to optimize hardware resource consumption.

5) EGURM Addressing. As shown above, EGURM is multifaceted and needs up to 8 parameters for addressing. Before addressing we should also decide which structure will be used – parallel or tree like.

6) Number of Operations and Hardware Resources. In the case of MA-based algorithm, we should choose how to

implement the rotator – using memory or without it. This decision depends on hardware resource consumption in general. Constant multiplication does not usually need multipliers. The choice of FPGA device family also influences the consumption of resources because of different relative proportions between a number of logic cells and built-in multipliers.

D. Accuracy of Algorithm

Requirements for the accuracy depend on application. The main source of inaccuracy is limited wordlengths (also the number of iterations for CORDIC) and rounding modes used inside the rotator. In [17] the detailed results are presented about estimation of errors for real transforms like (4). We can adopt the provided results for unitary transforms too, but that is a separate task for the future. In the tool presented in [15] a simple mean square error has been used for the control and comparison of results obtained in MATLAB and hardware.

*1) Fixed Point Arithmetic***.** At this moment, we consider the use of fixed point arithmetic (FPA) only – mainly because of availability of built-in multipliers in FPGA more suitable for the intensive use of FPA. We suggest also that the best, from the viewpoint of rotation algorithm, is the Q1.X FPA format (1 sign bit and X fraction bits). Such a choice is justified by the nature of rotation – vector length does not change after rotation and we can use the numbers normalized to 1. The second reason for the chosen format is a simpler bit cutting than in the case of integer numbers.

2) Wordlength. It is of great importance for a compromise between accuracy and hardware consumption. It is well known that after multiplication we have the doubled number of input bits, but after summation – one additional bit. In the case of implementation of, for example, (14), the output wordlength grows dramatically because of the cascade of two multiplications. Thus we cannot avoid the cutting of bits inside the rotator to minimize the hardware resources. In the case of complex CORDIC algorithm we should use 3 real CORDIC blocks. The choice of iteration number and the angle accumulator wordlength is also important in the context of accuracy.

V. Conclusion

This paper aims to highlight the problems that arise during the development of tools for automation of HDL code generation for devices based on parametrical unitary transforms. Some of the problems we solve in [15], where we report about the successfully developed automated tool for VHDL code generation of EGU-rotator.

VI. Acknowledgments

This work has been partly supported by the project 09.1552 (the Council of Science of Latvia) and by the National Program (the Ministry of Education and Science). We are also thankful to Andra Martinsone for collaboration and assistance.

References

[1] R. P. Brent and F. T Luk, "The solution of singular value and symmetric eigenvalue problems on multiprocessor arrays", SIAM J. Sci. Statistic Comput. vol. 6, pp. 69-84, 1985.

[2] J. Gotze, S. Paul, M. Sauer, "An efficient Jacobi-like algorithm for parallel eigenvalue computation", IEEE Transactions On Computers, Vol. 42, No. 9, pp.1058-1065, Sept. 1993.

[3] F. Lorenzelli and K. Yao, "SVD Updating for Nonstationary Data", IEEE Catalog Number 0-7803-2123494, 1994, pp.450-459.

[4] USA patent US20060155798, "Eigenvalue Decomposition and Singular Value Decomposition of matrices using Jacobi rotation".

[5] Weiwei Ma et al, "An FPGA-based Singular Value Decomposition processor", IEEE Catalog No. 1-4244-0038-4 2006, 2006, pp. 1047-1050.

[6] I. Bravo et al, "Implementation in FPGA of Jacobi method to solve the eigenvalue and eigenvector problem", IEEE Catalog No. 1-4244-0 312-X/06, 2006, 4 pages.

[7] T. Wang, P. Wei, "Hardware efficient architectures of improved Jacobi method to solve the eigen problem", IEEE Catalog No. 978-1-4244-6349-7/10, 2010. pp. 22-25.

[8] M. Kim, K. Ichige, H. Arai, "Implementation of FPGA based fast DOA estimator using unitary MUSIC algorithm", IEEE Catalog No. 0-7803-7954-3/03, IEEE, 2003, pp.213-217.

[9] Y-R. Lin, J. Mar, "Realization of subspace-based digital beamformer using CORDIC algorithm based on Software Defined Radio architecture", IEEE Catalog No. 978-1-4244-5213-4/09, 2009, 5 pages.

[10] M. Otte, M. Buckes and J. Gotze, "Complex CORDIC-like algorithms for linearly constrained MVDR beamforming", IEEE Catalog No. 0-7803-5977-1/00/, 2000 IEEE, 8 pages.

[11] R. Prain, A. Paplinski, "A distributed arithmetic online rotator for signal processing applications", IEEE Catalog No. 0-7695-2203-3/04, 2004, 8 pages.

[12] G. Valters, P. Misans, "FPGA implementation of Elementary Generalized Unitary Rotation", the 27th IEEE Norchip 2009 Conference, Trondheim, Norway, Nov. 16 17, 2009, in Proc. (on flash memory) of the Conf., ISBN 978-1-4244-4311-6/09, 2009, 4 pages.

[13] R. Andraka, "A survey of CORDIC algorithms for FPGA based computers". Proc. of the 1998 ACM/SIGDA 6th Int. Symposium on FPGA, Feb. 22-24, 1998, Monterey, CA. pp. 191-200.

[14] P. Misans, G. Valters, "Initial FPGA design for Generalized Orthogonal Nonsinusoidal Division Multiplexing". The 27th IEEE Norchip 2009 Conference, Trondheim, Norway, Nov. 16 17, 2009, in Proc. (on flash memory) of the Conf., ISBN 978-1-4244-4311-6/09 2009, IEEE, 5 pages.

[15] G. Valters, "Initial version of MATLAB/ SIMULINK based tool for VHDL code generation and FPGA implementation of Elementary Generalized Unitary Rotation", in press, 6 pages.

[16] http:\\www.ieee.org\ieeexplore.

[17] M. Terauds, "A new kind of discrete orthogonal transforms and errors in the corresponding signal processing applications", the PhD thesis, Faculty of Electronics and Telecommunications, Riga Technical University, 2009.

[18] 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?", The Nordic MATLAB User Conf. 2008, Stockholm, Sweden, Nov. 20.-21., 2008, pp. 1-8.

Gatis Valters was born in Latvia on May 18, 1984. He received the B.sc.ing., M.sc.ing. degrees from Riga Technical University, Latvia, in 2005 and 2007, respectively. Currently he has been working towards obtaining the Doctoral degree in Electronics. In 2007, he joined the Department of Fundamentals of Electronics, Riga Technical University, as an Assistant. His main areas of research interest are FPGA-based DSP.
Email: name.urname@rtu.lv

Peteris Misans is a Professor of Riga Technical University. He received the Doctoral degree in Electronics and Telecommunications in 1985 from Riga Technical University. His recent research interests are novel unitary discrete transforms, their implementations and applications.

Gatis Valters, Pēteris Misāns. Uz elementārās vispārinātās rotācijas balstītu unitāru pārveidojumu automatizācijas īstenošana FPGA mikroshēmās
Rakstā ir aprakstītas kompleksās Jakobi rotācijai līdzīgās elementārās vispārinātās unitārās rotācijas (EGU-rotācijas) un parametrisko unitāro pārveidojumu, kas izmanto EGU-rotācijas, veidošanās pamati. EGU-rotācijas matricai (EGURM) ir 64 dažādi veidi un vairāki simti modifikāciju dažādiem leņķu komplektiem. EGURM adresācijai ir nepieciešami līdz pat 8 parametri. Kopējais EGURM skaits var sasniegt vairākus tūkstošus, kas loģiski noved pie pārveidojumu sintēzes automatizācijas nepieciešamības. Detalizētāk tiek aplūkots uz reizinātājiem un summatoriem balstīts rotācijas algoritms, kas ir piemērotāks ātrdarbīgiem pielietojumiem. Eksistē 7 EGURM modifikāciju kategorijas, kurās vienas EGU-rotācijas īstenošanas ietvaros reizinātāju un summatoru skaits mainās plašās robežās. Ir analizēts nepieciešamo rotāciju skaits gan paralēlās, gan kokveida struktūras pārveidojumu īstenošanai. Par optimālu (no operāciju skaita viedokļa) ir atzīta pārveidojuma kokveida struktūra. Tiek parādīts, ka parametriskajiem pārveidojumiem EGURM mainās laikā, bet vislielākā rotāciju dažādība ir pirmajās/pēdējās dekompozīcijas/rekonstrukcijas pakāpēs. Ir noskaidrots, ka, ja īstenojot pārveidojumus ir nepieciešams minimizēt izmantojamo loģisko elementu skaitu, saglabājot ātrdarbību, pirmajās/pēdējās (dekompozīcijai/rekonstrukcijai) kaskādēs vēlams izmantot uz reizinātājiem un summatoriem balstītu rotāciju, bet pārējās kaskādēs vajag izmantot CORDIC algoritmu. Ir akcentēti galvenie uzdevumi, kas ir jārisina, veidojot automatizētu sistēmu. Veidojamajai sistēmai ir jāatbalsta paralēlās/kokveida arhitektūras izstrādi, simboliskās matemātikas un teksta apstrādes iespējas, dažādu algoritmu tipus, FPGA sintēzes procesā izmantojamo resursu novērtējumu. Bez tam sistēmā ir jābūt iespējai mainīt fiksētā punkta aritmētikas vārda garumu kontekstā ar resursu patēriņa un pārveidojuma precizitāti. Ir konstatēts, ka uz unitārajiem pārveidojumiem balstītu ierīču izstrāde ir sarežģīts uzdevums. Priekšā stādāmo darbu var uzskatīt par vadlīnijām atbilstošu automatizēto līdzekļu izstrādē.

Гатис Валтерс, Петерис Мисанс. Автоматизация реализации элементарной обобщенной унитарной ротации в FPGA
В статье описываются основы комплексной Якоби подобной элементарной обобщенной унитарной ротации (EGU-ротации) и основы построения параметрических унитарных преобразований, используя EGU-ротации. Матрица EGU–ротации (EGURM) является многоликой, и существует 64 разных видов и до нескольких сот разновидностей для разных комплектов, используемых углов поворота. Для адресации EGURM необходимо до 8 параметров. Общее число EGURM может достигнуть нескольких тысяч, что логически приводит к необходимости автоматизации синтеза преобразований. Из-за удобства в высокоскоростных применениях более детально обсуждается алгоритм, в котором ротация основывается на умножениях и суммированиях. Существует 7 категорий EGURM модификаций, в которых для реализации одной EGU–ротации число умножителей и сумматоров меняется в широких пределах. Проанализировано число ротаций, необходимых для реализации как паралельной, так и древовидной структуры преобразований. Оптимальной с точки зрения числа операций признана древовидная структура преобразования. Показано, что для параметрических преобразований EGURM меняется во времени, а наибольшая разновидность ротаций наблюдается в первых/последних каскадах декомпозиции/реконструкции. Выяснено, что если при реализации преобразований необходимо минимизировать число используемых логических элементов сохраняя быстродействие, то в первых/последних (декомпозиции/реконструкции) каскадах желательно использовать ротацию основанную на умножителях и сумматорах, а в остальных каскадах необходимо использовать CORDIC алгоритм. Акцентированы главные задачи, которые необходимо решать при создании автоматизированной системы. Создаваемая система должна поддерживать разработку паралельной/древовидной архитектуры, возможности снмвольной математики и текстовой обработки, разные типы алгоритмов и оценку ресурсов используемых в процессе синтеза FPGA. Система также должна иметь возможность менять длину слова для арифметики с фиксированной запятой в контексте с затратами ресурсов и точностью преобразования. Констатировано, что разработка устройств, основанных на унитарных преобразованиях, является задачей высокой сдожности. Представляемую работу можно рассматриватъ как основу путеводителя для разработки соответствующих средств автоматизации.