Hardware Acceleration of Maximum-Likelihood Angle Estimation for Automotive MIMO Radars

Frank Meinl, Martin Kunert
Advanced Engineering Sensor Systems
Robert Bosch GmbH
Leonberg, Germany
frank.meinl@de.bosch.com

Holger Blume
Institut für Mikroelektronische Systeme
Leibniz Universität Hannover
Hannover, Germany
blume@ims.uni-hannover.de

***Abstract*—Direction of arrival (DOA) estimation is an important array signal processing technique, used by various applications such as radar, sonar or wireless communication. Most of the known DOA algorithms suffer from a significant performance reduction and even fail completely under difficult conditions, like small antenna aperture size, correlated signals or a small number of snapshots. Maximum-Likelihood (ML) methods have been investigated thoroughly and are known to still work even in such difficult scenarios. Though, the major drawback of ML methods is their computational cost, especially in the case of large MIMO (multiple-input multiple-output) configurations. This work presents a novel hardware accelerator architecture, which is able to compute the exact ML estimation in the case of one or two targets. It is shown, that the computational demanding vector product can be implemented with the help of CORDIC units, which help to save a considerable amount of hardware resources. Furthermore, the result of the single target estimator can be reused to efficiently compute the estimates in the two-target case. Finally, the performance of the architecture is evaluated by a FPGA implementation which is able to process more than 20000 detections from 16 channels with 256 steering vectors in real-time (25 Hz).**

Keywords*—Automotive radar; driver assistance; direction of arrival; maximum-likelihood estimation; radar signal processing; CORDIC; hardware accelerator; FPGA

I. Introduction

The problem of estimating the direction of arrival (DOA) of an incident signal has been exhaustively investigated in the past and is relatively well understood. A major interest in DOA techniques arises certainly from the diverse radar and sonar sensor technologies which rely on a precise localization of one or multiple sources. The variety of radar and sonar applications, whether civilian or military, range from small embedded sensors to fixed radar stations with huge antennas. Many different approaches to DOA estimation can be identified, for instance triangulation, the use of antenna arrays or mechanically steerable antennas. Beside this specific localization technology, the usage of DOA techniques for wireless communication gained more and more attention in the past few years [1]. These applications benefit from beam-forming techniques, which can reduce channel interference between multiple users. For this reason, the position of mobiles and thus the DOA estimation is of great interest.

In addition to the well-established radar technology present in most of today's airplanes and ships, new sensor concepts have been recently developed, appropriate for the operation in civilian vehicles on public roads. Advanced driver assistance systems (ADAS) as well as the emerging trend of autonomous-driving cars even strengthen this progress due to their need for accurate and reliable sensor technology. By now, radar sensors play a key role for many assistance functions owing to their robust and accurate measurement principle. The widely used frequency-modulated continuous-wave (FMCW) modulation technique provides an exact range measurement combined with an instant measurement of the velocity (Doppler). In combination with a DOA estimation, the objects in the vicinity of the sensor can be detected, separated and classified. However, a further increase in resolution will become necessary in the near future to handle even complex and difficult situations on the way to automated driving.

Automotive radar sensors are heavily constrained in size, cost, emitted power and measurement time. In addition, multiple targets are present and the environment is clearly non-static which makes the DOA estimation even more difficult in this case. In order to resolve closely spaced objects, a high angular separability is required which in turn is heavily dependent on the antenna aperture size of the sensor. At the same time, the extent of the sensor is strictly limited by mechanical and design constraints of the vehicles. To overcome this problem, high resolution angle estimation algorithms can be used, but in general they impose a higher computational burden to the signal processing unit. Further improvements can be based on more sophisticated antenna array designs which have a greater number of sending or receiving channels. The usage of MIMO arrays has attracted more attention recently because it can increase the number of virtual channels with a manageable hardware effort. Again, the processing of a great number of channels in real-time imposes higher requirements to the processing unit. For this reason, the employment of highly efficient accelerators and ASICs for radar-based ADAS is foreseeable in the near future [2], [3].

A major drawback for high-resolution DOA estimation in the case of automotive radars is the number of snapshots, i.e. the number of independent measurement cycles, which is generally small due to the non-stationarity of the environment. Especially when the data is subject to a range-Doppler preprocessing, like it is the case for many radar applications, only a single snapshot is available. This impedes the usage of many DOA methods and justifies the usage of computational demanding maximum-likelihood (ML) estimators.

ML estimators belong to a special class of DOA algorithms with some important statistic properties. They are efficient in the sense that they achieve the lower Cramer-Rao bound for large sample sizes. In addition, it has been shown that the performance for small sample sizes or even in the single snapshot case is superior to other estimators [4][5]. Though, the immediate use for real applications was inacceptable for a long time due to their high computational cost. In most cases, ML estimators need to find the global maximum of a non-linear, multidimensional function. In general, global search procedures are required, because no closed form solution for this optimization problem exists. Iterative search strategies have been reported, however their convergence is highly dependent on the initial value [6].

In general, the complexity grows with the number of dimensions, i.e. the number of signal sources, which lead to the development of approximate methods. The advantage of many algorithms is that they only require a 1D search, for instance the minimum-variance (Capon [7]) or MUSIC estimator [8]. Another class of estimators exploit the regularity of a uniform linear array (ULA) whose antenna elements are all equally spaced. For instance ESPRIT [9] or Root-MUSIC [10] are prominent examples which exploit such properties of the array design. A lot of those algorithms have in common that they require the number of snapshots to be greater than the number of sources. In addition, if the incident signals are correlated, the performance degrades heavily and in the case of fully coherent signals the estimation may even completely fail. In contrast, the performance of ML methods is still satisfactory, which makes them essential for certain applications. Furthermore, they still work if only a single snapshot is available. For this reason, ML methods are very attractive for the usage in automotive radars, which made them subject to research [5][11]. More background information about DOA estimation methods can be found in [12] or [14].

In the following work, solutions for accelerating ML DOA estimation are presented. The large number of channels which can be attained with MIMO antenna arrays is taken into account by the proposed architecture. Section II describes the theory of DOA estimation and the ML estimation. A brief introduction to the operating mode of the CORDIC processor and its application for computing the scalar product is given in section III.B. In the following sections III.C and III.D a hardware accelerator architecture is presented, capable of estimating the angle of one or two targets while processing a high number of channels in real time. Different implementation options are evaluated in terms of numerical accuracy and resource usage. This architecture is implemented and evaluated on a Xilinx Virtex-7 FPGA and compared against a vectorized CPU implementation. The results are presented in section IV.

II. Deterministic Maximum-Likelihood (DML) Estimation

A. *Direction of arrival estimation*

A plane wave impinging on a sensor array with M elements generates a certain amplitude and phase response, depending on its direction of arrival (DOA). The measured array output vector $\boldsymbol{x}$ contains consequently M entries and can be used to estimate the DOA(s) of one or multiple sources. For this purpose it is convenient to model the array response in the case of an ideal, single source under each possible incident angle θ. The resulting array output is referred to as steering vector $\boldsymbol{a}(\theta)$.

The steering vector can be either derived theoretically from the array geometry or it can be measured with an optimized calibration setup, where a single target under a known angle θ is present. If the received power is equal for all array elements, i.e. the channel gain is the same for all elements, then the steering vector can be written in the following form, where the phase ϕ is normalized to the first channel (cf. Fig. 1).

$$\boldsymbol{a}(\theta) = \begin{bmatrix} 1 & e^{i\phi_2(\theta)} & \cdots & e^{i\phi_M(\theta)} \end{bmatrix} \tag{1}$$

The signal model which is used in the following assumes that K complex signals s_k impinge on the antenna array. The signals can be integrated into the signal vector $\boldsymbol{s} = [s_1 \quad s_2 \quad \cdots \quad s_K]^T$ and the array output in this case is a superposition of the K signals. Finally, a noise vector $\boldsymbol{n}$, whose elements are all complex Gaussian distributed and independent (i.i.d.), is added to the measured array output $\boldsymbol{x}$:

$$\boldsymbol{x} = \boldsymbol{A}(\boldsymbol{\theta})\boldsymbol{s} + \boldsymbol{n} \tag{2}$$

The matrix $\boldsymbol{A}(\boldsymbol{\theta})$ consists of the steering vectors $\boldsymbol{a}(\theta_1), \boldsymbol{a}(\theta_2), \ldots, \boldsymbol{a}(\theta_K)$ which relate to the signals $\boldsymbol{s}$.

The problem to solve consists of two steps: The number of sources K has to be estimated. Afterwards, an incident angle θ has to be found for each source s_k. For the following derivation in this work, the number of sources is assumed to be known. In practice, solutions to the problem of order estimation have been investigated in-depth and are not in the scope of this publication (cf. [12] p. 1054, [13]). The focus lies only on the determination of the incident angles θ_k in the case of one or two targets.

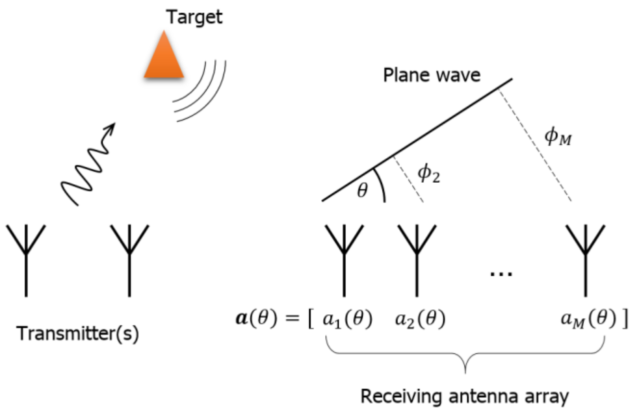
Target
Plane wave
ϕ_M
ϕ_2
θ
Transmitter(s)
$\boldsymbol{a}(\theta) = [\ a_1(\theta) \quad a_2(\theta) \quad \ldots \quad a_M(\theta)\ ]$
Receiving antenna array

Fig. 1. A plane wave impinging on an antenna array with M elements.

B. *Deterministic Maximum-Likelihood (DML) Estimation*

The signal models often distinguish two different variants of ML estimation. The first one assumes a deterministic signal, which is often present in communication or active radar applications where the waveform is known. The second one adopts to a non-deterministic signal where the signal to detect has not been identified yet, e.g. when passively locating an unknown sender. This leads to two different results of which the first is known as conditional ML estimation (CMLE) or deterministic ML (DML). The second one is referred to as unconditional ML estimation (UMLE) or statistical ML (SML). This paper is entirely based on DML estimation, because the received signals result from echoes of the transmitted and hence known waveform.

The DML function $\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{s})$ for the signal model of (2) in the case of a single snapshot is given by the following expression [12]:

$$\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{s}) = \frac{1}{\det[\pi\sigma^2 \boldsymbol{I}]} \exp\left(-\frac{1}{\sigma^2}|\boldsymbol{x} - \boldsymbol{A}(\boldsymbol{\theta})\boldsymbol{s}|^2\right) \tag{3}$$

where σ is the variance of the Gaussian noise. Applying the natural logarithm function results in the log-likelihood function, which becomes maximal for the same argument values of $\boldsymbol{\theta}$, however the computation is simplified.

$$\ln \mathcal{L}(\boldsymbol{\theta}, \boldsymbol{s}) = -K \ln \sigma^2 - \frac{1}{\sigma^2}|\boldsymbol{x} - \boldsymbol{A}(\boldsymbol{\theta})\boldsymbol{s}|^2 \tag{4}$$

The goal is to maximize this function which can be done by fixing all but one of the parameters and forming the partial derivation with respect to the arbitrary parameter. For the reason of brevity the solution is just given as is, a complete derivation can be found in [12], pp. 1004. The dependency of $\boldsymbol{A}$ to the parameter $\boldsymbol{\theta}$ has been omitted to simplify the expression.

$$\underset{\boldsymbol{\theta}}{\operatorname{argmin}}\ |\boldsymbol{x} - \boldsymbol{A}(\boldsymbol{A}^H\boldsymbol{A})^{-1}\boldsymbol{A}^H\boldsymbol{x}|^2 \tag{5}$$

The minimization problem can be rewritten as

$$\underset{\boldsymbol{\theta}}{\operatorname{argmax}}\ \boldsymbol{x}^H\boldsymbol{A}(\boldsymbol{A}^H\boldsymbol{A})^{-1}\boldsymbol{A}^H\boldsymbol{x} \tag{6}$$

In general, no closed form solution exists for this expression, which is why an exhaustive maximum search has to be performed. In [4], an approximate estimation method for two closely spaced targets has been developed, which is however only valid if the targets are known to differ only in a small angle $\Delta\theta$. For this reason, this approximation cannot be used since the values of $\Delta\theta$ are not constrained and can take arbitrary values in the case of automotive radar sensors.

For a single target, the matrix $\boldsymbol{A}^H\boldsymbol{A}$ has only one entry and thus the inverse is just the reciprocal value. Furthermore, if the norm of the steering vector is known to be equal one (which is no restriction in general), the computation of the inverse gets trivial and (6) simplifies to the following expression:

$$\underset{\theta}{\operatorname{argmax}}\ |\boldsymbol{x}^H\boldsymbol{a}(\theta)|^2 = \underset{\theta}{\operatorname{argmax}}\ |r_\theta|^2 \tag{7}$$

In the equation above, the scalar product between the measured vector $\boldsymbol{x}$ and the array steering vector $\boldsymbol{a}(\theta)$ is denoted as r_θ for convenience.

For two targets, the inverse matrix of $\boldsymbol{A}^H\boldsymbol{A}$ has to be computed. This can be done by using the simple analytic solution for a 2x2 matrix. Let the steering vectors $\boldsymbol{a}(\theta_i)$ and $\boldsymbol{a}(\theta_j)$ be denoted as $\boldsymbol{a}_i$ and $\boldsymbol{a}_j$, respectively. Then the matrix $\boldsymbol{A}^H\boldsymbol{A}$ and its inverse can be written as follows:

$$\boldsymbol{A}^H\boldsymbol{A} = \begin{bmatrix} \boldsymbol{a}_i^H \\ \boldsymbol{a}_j^H \end{bmatrix} [\boldsymbol{a}_i \quad \boldsymbol{a}_j] = \begin{bmatrix} |\boldsymbol{a}_i|^2 & \boldsymbol{a}_i^H\boldsymbol{a}_j \\ \boldsymbol{a}_j^H\boldsymbol{a}_i & |\boldsymbol{a}_j|^2 \end{bmatrix} = \begin{bmatrix} 1 & \beta_{ij} \\ \beta_{ij}^* & 1 \end{bmatrix} \tag{8}$$

$$(\boldsymbol{A}^H\boldsymbol{A})^{-1} = \frac{1}{1 - |\beta_{ij}|^2} \begin{bmatrix} 1 & -\beta_{ij} \\ -\beta_{ij}^* & 1 \end{bmatrix} \tag{9}$$

Combining (6) and (9) results in a maximization problem for the two target case:

$$\underset{\theta_i,\theta_j}{\operatorname{argmax}} \frac{1}{1 - |\beta_{ij}|^2} [r_i \quad r_j] \begin{bmatrix} 1 & -\beta_{ij} \\ -\beta_{ij}^* & 1 \end{bmatrix} \begin{bmatrix} r_i^* \\ r_j^* \end{bmatrix} \tag{10}$$

With some simplifications, this leads to the following expression for the DML estimator in the case of two targets [11]:

$$\underset{\theta_i,\theta_j}{\operatorname{argmax}} \frac{|r_i|^2 + |r_j|^2 - 2\mathrm{Re}[r_i r_j^* \beta_{ij}]}{1 - |\beta_{ij}|^2} \tag{11}$$

Where the operator Re[·] represents the real part of a complex number. Cleary, the complexity of computing one value of the ML function is more than three times as expensive as in the single target case, because three scalar products (r_i, r_j and β_{ij}) have to be computed. In addition, the number of possible angle values is squared, which is even more severe in the case of a full grid search. In the following section, an implementation of the DML estimator incorporating several optimizations is presented which makes a real-time computation nevertheless possible.

III. Hardware Architecture

A. *Scalar product*

The evaluation of the ML function depends heavily on the calculation of the scalar product between the measured array vector and the array steering vector. Both vectors contain complex numbers and thus each element involves 4 multiplications. Evidently, the complexity of this step grows linearly with the array size. For large MIMO arrays and in the case of a one-dimensional MLE, this step even dominates execution time. The following maximum search is not significant because it is independent of the number of channels. The optimization and acceleration of the scalar product computation is reasonable and some specific properties of the array steering vectors can be exploited.

If the individual channel gains are considered to be equal, all entries of the array steering vectors have the same amplitude. Thus, it is valid to only rotate the measured vector $\boldsymbol{x}$ without changing its amplitude.

In general, a vector rotation is not less demanding, because the complex numbers are represented as real and imaginary part and a complex multiplication is required anyway. However, the rotation can be realized efficiently in hardware by a single CORDIC unit, which is only built out of adders and is able to rotate a complex number by an arbitrary angle. The name CORDIC is an acronym for "coordinate rotation digital computer" [15].

B. Cordic

Similar to a multiplication by ten in the decimal system, certain arithmetic operations become trivial in the binary system. A multiplication by 2^n is simply conducted by shifting the binary number n bits to the left. Certain mathematical functions or arithmetic calculations can be efficiently implemented exploiting this simple property. CORDIC algorithms belong to the class of shift and add algorithms and can be used for the calculation of trigonometric, hyperbolic or logarithmic functions [16]. In fact, the idea behind CORDIC is to approximate a vector rotation by multiple subsequent smaller rotations, which can exploit multiplications with a power of two. Rotating a complex number $a + jb$ by a certain angle ϕ is performed by applying the following rotation matrix:

$$\begin{bmatrix} a' \\ b' \end{bmatrix} = \begin{bmatrix} \cos\phi & -\sin\phi \\ \sin\phi & \cos\phi \end{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix} = \cos\phi \begin{bmatrix} 1 & -\tan\phi \\ \tan\phi & 1 \end{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix}$$

If the $\tan\phi$ components are chosen equal to 2^{-n}, then only bit shift and add operations are necessary. The idea behind CORDIC is to split the rotation into several micro-rotations which then can be performed with simple additions. After each step, a decision for the subsequent rotation direction has to be taken. This algorithm converges linearly to the target rotation, i.e. the accuracy is about one bit per iteration.

Evidently, the amplitude of the vector is changed with each micro-rotation because the rotation matrix is not normalized anymore. It can be corrected by applying the correction factor $\cos\phi$ after each rotation. Alternatively, the accumulated scaling factors can be applied in the end to the final result. In this case, the total scaling factor converges to 0.6073 for increasing values of n, which corresponds to a gain of 1.647 [16]. However, if the absolute value of the ML function is scaled by a constant c, the argument of the resulting maximum is not affected, so that a compensation is not required at all, i.e.:

$$\underset{\theta}{\operatorname{argmax}} \ \ln \mathcal{L}(\boldsymbol{\theta}, \boldsymbol{s}) = \underset{\theta}{\operatorname{argmax}} \ c \ln \mathcal{L}(\boldsymbol{\theta}, \boldsymbol{s})$$

If the CORDIC processor has to execute arbitrary rotations by an angle ϕ, which is known as rotation mode, the next rotation direction has to be found after each step. To compute this decision, an extra hardware unit has to be implemented which compares the actual angle to the target angle ϕ. However, if the angle of rotation is fixed, i.e. known in advance, the coefficients σ can be precomputed offline. This is the case for the implementation of the MLE, because the steering vectors are fixed for a given sensor. The used CORDIC implementation in this work is optimized for fixed angles with precomputed coefficients. As a result, the hardware effort for the decision module can be saved.

In the case of a fully pipelined implementation, the number of bit shifts between each CORDIC stage remains fixed and thus the operation can be hardwired. A dedicated logic circuit, such as a barrel shifter is not required at all. In Fig. 2, the resulting architecture is presented in the case of three CORDIC stages. The pipeline length grows with the word length n, because more CORDIC iterations become necessary. Two n-bit adders per stage are used, which can either add or subtract the real and imaginary values of the previous stage. The type of operations is selected by the precomputed coefficients σ. The bit shifts are illustrated as boxes for better understanding, even though they require no hardware resources. Such a high performance implementation produces one result per clock cycle and is the first choice when a high data throughput is desired.

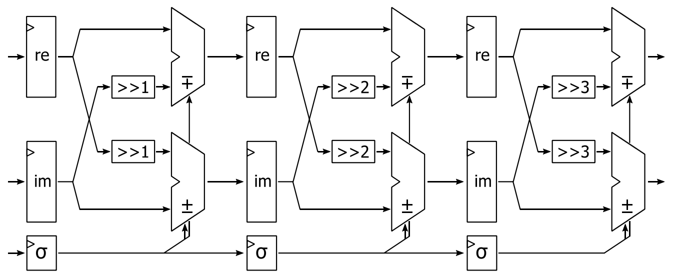

Fig. 2. Fully pipelined CORDIC implementation with three stages

Another optimization, suitable for fixed angle rotation would be the usage of a mixed radix CORDIC algorithm, where the last n/2 bits are computed with Radix-4 stages [17]. The total number of micro-rotations can be reduced by about 25 % and consequently the pipeline length decreases so that a considerable amount of logic can be saved. The more expensive determination of rotation factors for the Radix-4 case can be precomputed offline as well and does not engender any additional resource usage for the target hardware. If the full pipeline is implemented with Radix-4 stages, the total number of iterations could be even reduced by 50 % [18]. Though, due to the varying rotation step sizes, the scaling of the absolute value depends on the angle value and will be different for each vector element. As a result, an amplitude compensation has to be applied and the savings in the CORDIC unit itself would be diminished. However, when the individual channels have different gains and an amplitude compensation is needed anyway, the employment of pure Radix-4 CORDIC units could be beneficial.

C. Single target DML estimator

The main part of the resources is clearly occupied by the scalar product computation. The elements are summed up with the help of an adder tree, which is fully pipelined to maximize performance. The subsequent computation of the absolute squared value requires two multipliers and the following maximum search needs one comparator. These two steps are independent of the number of channels and do not contribute significantly to the total resource usage for large MIMO arrays which is mainly dominated by the scalar product. Depending on the comparison result, the absolute squared value is stored in a register which contains the final maximum value after all possible angles have been processed. An overview of the architecture is shown in Fig. 3. Two options to compute the complex scalar product are investigated. The conventional approach using binary multipliers and the optimized approach for vector rotations based on CORDIC processors.

The presented architecture is fully pipelined and computes the value of the DML function for one distinct angle. The final result is valid after the whole search space has been run through and depends thus on the number of different steering vectors. Typical values for the number of angles range about a few hundreds. Practical results show that a single estimation unit running at a frequency of 200 MHz provides enough throughput for the average amount of data. Anyway, the number of detections to be processed during one cycle is always limited by the processing unit and a trade-off has to be found for real applications. A further increase of performance could be achieved by simply implementing the architecture in parallel. The coefficient RAM could be reused, while the rest of the hardware is fully occupied and cannot be shared.

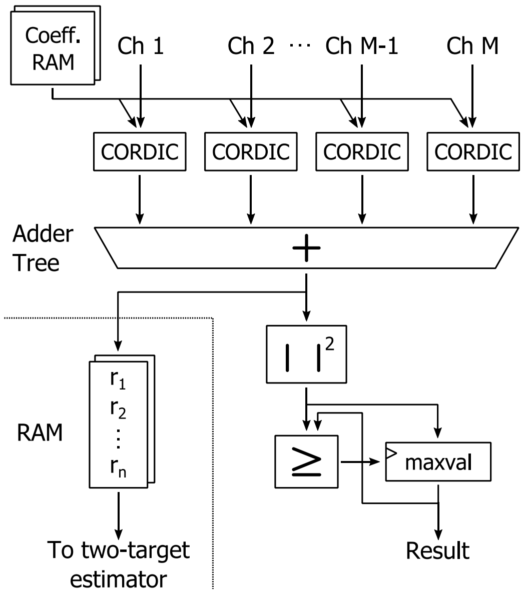

Fig. 3. Implementation of the single target estimator with the help of CORDIC units. The intermediate results are stored and reused by the two-target estimator.

D. Two-target DML estimator

The evaluation of the ML function itself requires the computation of three scalar products, r_1, r_2 and β_{12}, which dominate the complexity of this step. The straight forward approach is again to optimize multiplication with the help of CORDIC processors. Furthermore, if the computation of the two target case is following a single target estimation, the coefficients r_1 and r_2 have already been computed. The idea is to store all values r_θ in a small cache memory and then to reuse these coefficients. The last coefficient β is exclusively used in the two target case and depends only on the steering vectors. They are thus constant and can be precomputed offline, but a rather large look-up table is required due to many possible combinations of two different angles. The second option would be the usage of another array of CORDIC multipliers, which becomes quite expensive if many MIMO channels are present. For this work, a RAM based implementation has been chosen, but in practice this decision should be evaluated from case to case.

If the scalar products have already been computed and are stored in memory, as well as the coefficients, it remains to evaluate (11). In general, a two-dimensional grid has to be searched, however only the lower or upper triangle has to be evaluated because the ML function is symmetric with respect to $\theta_1$and θ_2. The module computing the two-target DML function is shown in Fig. 4. Similar to the single target case, it is able to output one value per clock cycle. Certainly, the whole grid search for one detection needs considerable more time than in the single target case because much more values have to be evaluated. This results from the two-dimensionality which squares the number of possible angles. Depending on the size of the intermediate RAM storing the r_θ values, the computation for the single-target DML function will be blocked at some point of time. A correct balancing between the one and the two target computation unit is required in order to achieve a high utilization of the hardware.

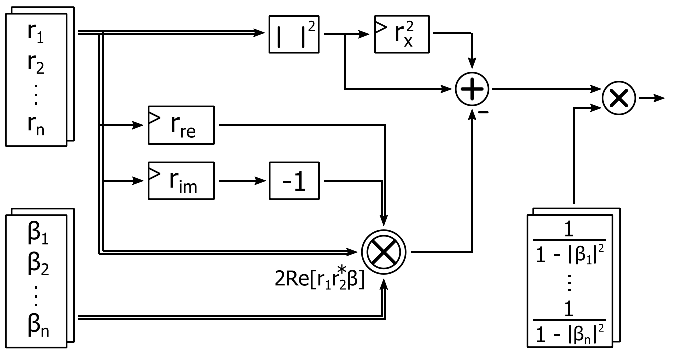

Fig. 4. Implementation of the two target estimator, corresponding to (11). The registers storing the values r_{re} , r_{im} , r_x^2 are loaded at the beginning of a new line.

It seems appropriate to compute multiple lines of the 2D matrix in parallel. This possibility is illustrated in Fig. 5. Each additional line has to wait one clock cycle after the neighboring line starts with the computation. Then, all lines operate on the same value of r_θ which is very efficient in terms of RAM load operations.

If a full 1:1 ratio balancing of one and two-target DML estimator is envisaged, the parallel evaluation of the two-target ML function would prevail the consumed hardware resources. Furthermore, the usage of the proposed parallel evaluation would be rather inefficient because the global utilization of all lines is expected to be 50 %. This results from the triangular shape of the data to be evaluated. An improved processing scheme, achieving a full utilization of all lanes is subject of further research.

In practice, if the number of targets is known beforehand, the computation of the two-target estimator can be skipped if just a single target is present. Consequently, if only a fraction of all radar detections require a two-target estimation, the implementation does not require that the computation for two targets finishes at the same time as the single target estimator does. Combined with an adequate buffer size between the two processing units, a high utilization of both modules becomes possible.

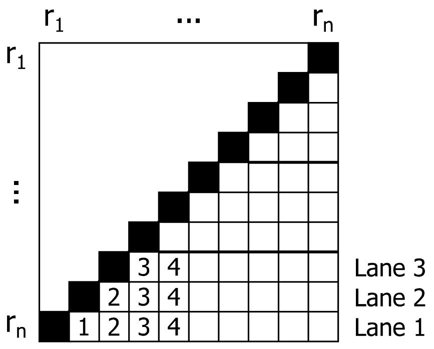

Fig. 5. Parallelization scheme of the two target estimator, where the numbers represent the current clock cycle. After two preloading cycles, all lanes operate on the same value of r_x.

IV. Results

The proposed architecture from the previous section has been implemented on a Xilinx Virtex-7 FPGA (XC7VX485T) to prove its effectiveness. Different word lengths have been evaluated as well as different architectural choices. In addition, the fixed point accuracy of the calculated scalar product has been evaluated with the help of simulated data and real measurements.

The provided numerical implementation results are intended to be used in future development of custom FPGA-designs or highly integrated ASICs. They set a first basis for future benchmarks and give the designer an idea about the trade-off between numerical accuracy and resource usage. Furthermore, the performance gain of the accelerator compared to a general purpose CPU is provided to illustrate its effectiveness.

A. Resource usage

The resource usage of the single target estimator is presented in TABLE I. It is mainly dominated by the scalar product which has been implemented in two different variants, of which the first one consists of CORDIC units and the second one is based on multipliers. Due to the availability of dedicated DSP blocks, which include optimized multiplier structures, the usage of LUTs is considerably lower when using these building blocks. For a better comparison, a third variant, namely a LUT based multiplier implementation is also presented. Even though the practical relevance is questionable for FPGA-based designs, the figures are better comparable to the CORDIC-based implementation in terms of logic usage and can be useful for custom hardware designs.

Interestingly, the CORDIC-based implementation requires only about half of the BRAM resources as a comparable multiplier-based implementation. The embedded RAM blocks are used as a coefficient storage and grow thus with increased accuracy. For the multiplication, it is necessary to store the real and the imaginary parts of the steering vector elements while in the case of the CORDIC implementation, only the coefficients of each micro-rotation have to be stored. These coefficients achieve the same accuracy with about half the word length and are thus more economical in terms of RAM usage.

TABLE I. Comparison of the resource usage for different implementation variants of the single target DML module. All values comprise 16 channels.

	Accuracy	LUT	Register	DSP	BRAM 18K
CORDIC	16 bit	9965	10316	4	8
	20 bit	15013	15432	4	10
	24 bit	21086	21572	8	12
DSP-based Multiplier	16 bit	544	562	68	15
	20 bit	6583	5898	68	18
	24 bit	2319	4418	136	22
LUT-based Multiplier	16 bit	20113	8361	4	15
	20 bit	32738	10403	4	18
	24 bit	46067	12444	8	22

The requirements of the CORDIC implementation is considerably lower as the multiplier-based version. This effect results mainly from the optimized implementation for a pure vector rotation. The used multiplier implementation is able to perform arbitrary multiplications, but this capability is not exploited in the first place. Evidently, if the elements of the steering vectors differ in magnitude and a scaling becomes necessary, the advantage of the CORDIC implementation would decrease or even disappear. Further optimizations for the multiplier implementation are also possible, for instance the realization of a complex multiplication by only three multipliers or the employment of booth multipliers. At the same time, the CORDIC implementation can also be improved, e.g. by the usage of Radix-4 stages, as already mentioned in section III.B.

In contrast to the single target estimation module, which includes the computation of the scalar product, the two target estimation module does not depend on the number of channels. The engendered resource usage is presented in TABLE II. It depends mainly on the number of steering vectors and the number of parallel lanes.

TABLE II. Comparison of the resource usage of the two-target DML module with different degrees of parallelization

	Accuracy	LUT	Register	DSP	BRAM 18K
1 Lane	16 bit	164	254	8	108
	20 bit	775	769	10	108
	24 bit	409	593	16	108
4 Lanes	16 bit	567	1071	26	108
	20 bit	2497	2703	34	108
	24 bit	1406	2220	52	108
16 Lanes	16 bit	1965	4296	98	108
	20 bit	9192	10429	130	108
	24 bit	5196	8746	196	108

B. Fixed-point accuracy

In terms of fixed-point accuracy, both realizations have been evaluated with the help of real measurement data. The error engendered by the fixed word length has been measured for the final value of the scalar product, i.e. after the summation of all elements. The result is then compared to a software implementation using double precision floating point values.

The numerical error is expressed in terms of peak signal-to-noise ratio (PSNR), i.e. the ratio between the squared full scale value and the mean square error. For a lower number of channels, the accuracy of the multipliers are superior, which can be justified by the different truncation behavior. In the case of a multiplication, the full width result is computed first, and the result is only truncated once in the end. The CORDIC-implementation truncates the result after each stage, i.e. multiple times and so each time a small amount of quantization noise is accumulated. However, this effect is reduced if more channels are used and thus more independent elements are added in the final summation. It seems that even though the amount of quantization noise of a single result is larger when a CORDIC-implementation is used, it gets equalized in the total result if more elements are summed up. Even though measureable, the effect is very small and in practice the word length can be adjusted accordingly to the PSNR requirements of the application.

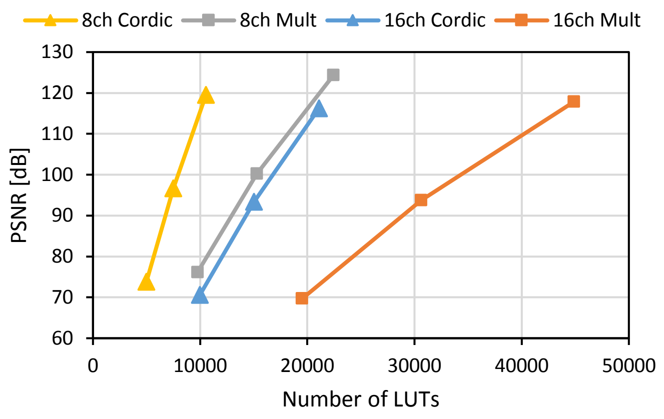

Fig. 6. Relationship between numerical accuracy and resource usage. Two different implementation variants (CORDIC and multipliers) and different number of channels *M* are compared.

C. *Performance*

The execution time of the hardware accelerator is highly dependent on the used clock frequency. Both modules have been designed to compute one ML estimate per clock cycle. Thus, the computation time per detection scales linear with the number of steering vectors n_a. The FPGA implementation has been realized with a clock frequency of 200 MHz which results in the performance data presented in the following. Nevertheless, a further increase of the frequency would be still possible along with a higher design effort.

The CPU-based reference implementation is an optimized computation kernel, exploiting the capability of 256 bit AVX vector instructions. The benchmarks have been run on an Intel i5 Ivy-Bridge CPU with 2.6 GHz with a TDP of 35 W. By reason of the rather limited fixed-point vector instructions, all computations are performed with the help of single or double precision floating point values. They are abbreviated as SP or DP, respectively. Furthermore, the implementation only makes use of a single core. In future work, the performance could be further increased by using multi-threading.

The speedup of the FPGA-based single target DML accelerator increases with the number of channels. This results from the parallel nature of the module, which applies all elements of the steering vector in parallel. In contrast, the CPU has to iterate over all channels and consequently, the required time depends on the number of channels. The execution time of the single target module is presented in Fig. 7 for two different numbers of steering vectors n_a. The speedup of the FPGA compared to the single precision implementation ranges from approximately 1.4x in the case of 8 channels up to a factor 14x for 64 channels. The estimated power dissipation reported by the tools is slightly below 1 W for 16 channels, regardless of the used implementation variant. This demonstrates the high efficiency compared to the CPU.

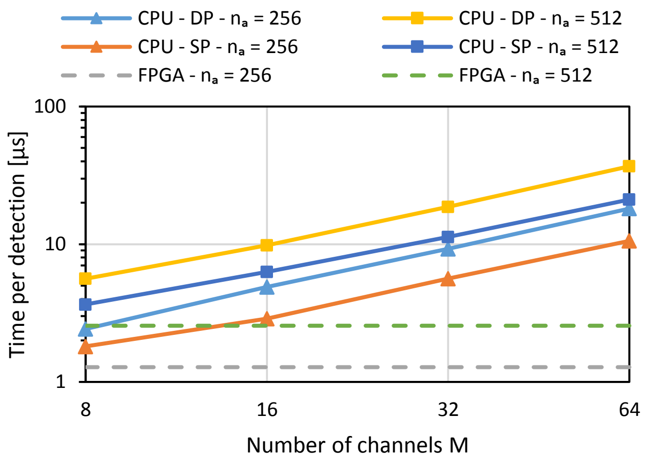

Fig. 7. Computation time per detection against number of channels for the single target estimator module. Different number of steering vectors n_a and different hardware platforms (CPU-DP, CPU-SP, FPGA) are compared.

The performance of the two target estimator is mainly dependent on the number of steering vectors, because the values of the scalar products r_θ have been computed prior to its invocation. Hence, there is no dependence on the number of channels as in the single target case. The effect of the parallelization for the FPGA module can be seen in Fig. 8.

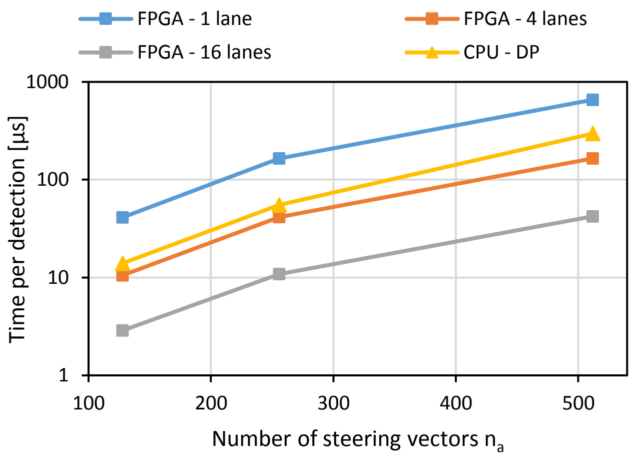

Fig. 8. Computation time per detection against number of steering vectors for the two target estimator module.

With 16 lanes in parallel, a speedup of 7x compared to the CPU version can be achieved. Compared to the single target estimator, the total time per detection is approximately one order of magnitude higher for $n_a = 256$ steering vectors. This performance gap further increases for larger n_a, so that an even higher degree of parallelization could become necessary in this case. The reported power dissipation for the two target module with 16 lanes in parallel is 1.2 W. Again, they are clearly below the specified values of the CPU.

V. Conclusion

The performance of the ML estimator is among the best performing DOA estimation methods, especially in the single snapshot case. However, the employment for time-sensitive applications is strictly limited due to the high computational requirements. A novel hardware accelerator architecture has been proposed in this paper, enabling a computation of the ML estimator in real-time, even for large MIMO antenna arrays. Unlike conventional processor based implementations which are built upon on the available instruction set, several properties of the ML computation can be exploited which leads to a powerful and efficient hardware realization. The proposed architecture has been successfully implemented on an FPGA in order to show its effectiveness. The obtained results in terms of resource usage, numerical accuracy and processing performance have been presented in this work.

The comparison of different implementation alternatives shows, that the employment of CORDIC modules for the calculation of the scalar product can save a significant amount of resources. In contrast to the LUT-based multiplier implementation, virtually half of the logic resources can be saved. Furthermore, the usage of a dedicated buffer memory helps to accelerate the estimation for the two target estimator. The processing time in the two target case can be further reduced by the presented parallelization scheme. It allows the computation of multiple results at the same time while the degree of parallelism can be freely chosen. This helps to build highly efficient accelerators which can be adapted according to the application's requirements. In comparison to a CPU-based reference implementation, the FPGA architecture for the two-target estimator with 16 parallel lanes offers a speedup of factor 7. At the same time, only about 5 % of the total FPGA's resources have been used, which demonstrates the efficiency of the architecture. The overall results indicate that ML-based DOA estimation is feasible with the proposed architecture, even for large MIMO configurations which comprise a considerable number of channels.

Further development of the system would incorporate the estimation of three or more targets. For these cases, specialized search strategies could become necessary as the search space increases exponentially and an exhaustive search is not possible anymore. Additionally, the order estimation could be tightly coupled with the ML estimation unit. In combination with a sequential hypothesis testing, the appropriate estimator can be calculated only when necessary.

Additional room for improvement is expected in the word length of intermediate results along the data path. By using proper word lengths for coefficients and results at different steps of the computation, a higher efficiency can be expected. The result would be a lower resource usage for the same level of PSNR.

References

[1] L.C. Godara. "Application of antenna arrays to mobile communications. II. Beam-forming and direction-of-arrival considerations." *Proceedings of the IEEE* 85.8 (1997): 1195-1245.

[2] Freescale Semiconductor, Inc. "Highly integrated automotive radar MCU for advanced driver assistance systems." *MPC577xK-MCU - Fact Sheet (REV 1),* 06/2013

[3] Infineon Technologies AG. "TC297TA – AURIX™ family. Dedicated to driver assistance systems." *Product Brief v01_00,* 08/2016

[4] F. Vincent, O. Besson and E. Chaumette. "Approximate maximum likelihood estimation of two closely spaced sources." *Signal Processing* 97 (2014): 83-90.

[5] P. Häcker and B. Yang. "Single snapshot DOA estimation." *Advances in Radio Science* 8.16 (2010): 251-256.

[6] I. Ziskind and M. Wax. "Maximum likelihood localization of multiple sources by alternating projection." *IEEE Transactions on Acoustics, Speech and Signal Processing* 36.10 (1988): 1553-1560.

[7] J. Capon. "High-resolution frequency-wavenumber spectrum analysis." *Proceedings of the IEEE* 57.8 (1969): 1408-1418.

[8] R.O. Schmidt. "Multiple emitter location and signal parameter estimation." *IEEE Transactions on Antennas and Propagation* 34.3 (1986): 276-280.

[9] R. Roy and T. Kailath. "ESPRIT-estimation of signal parameters via rotational invariance techniques." *IEEE Transactions on Acoustics, Speech and Signal Processing* 37.7 (1989): 984-995.

[10] A.J. Barabell. "Improving the resolution performance of eigenstructure-based direction-finding algorithms." *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP) Vol. 8,* 1983.

[11] P. Heidenreich. "Antenna array processing: autocalibration and fast high-resolution methods for automotive radar." *Ph.D. Thesis Darmstadt,* 2012.

[12] H.L. van Trees. "Detection, estimation, and modulation theory, Part IV: Optimum array processing." *John Wiley & Sons,* 2004.

[13] B. Ottersten, M. Viberg, P. Stoica and A. Nehorai. "Exact and large sample maximum likelihood techniques for parameter estimation and detection in array processing." *Springer Berlin Heidelberg,* 1993.

[14] H. Krim and M. Viberg. "Two decades of array signal processing research: the parametric approach." *IEEE Signal Processing Magazine* 13.4 (1996): 67-94.

[15] J.E. Volder. "The CORDIC trigonometric computing technique." *IRE Transactions on Electronic Computers* 3 (1959): 330-334.

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

[17] J.A. Lee and T. Lang. "Constant-factor redundant CORDIC for angle calculation and rotation." *IEEE Transactions on Computers* 41.8 (1992): 1016-1025.

[18] E. Antelo, J. Villalba, J.D. Bruguera and E.L. Zapata. "High performance rotation architectures based on the radix-4 CORDIC algorithm." *IEEE Transactions on Computers* 46.8 (1997): 855-870.