A Modified Decomposed Vector Rotation-Based Behavioral Model with Efficient Hardware Implementation for Digital Predistortion of RF Power Amplifiers

Wenhui Cao, *Student Member*, *IEEE,* and Anding Zhu, *Senior Member*, *IEEE*

***Abstract* — This paper proposes a novel hardware implementation strategy to achieve low-cost design for digital predistortion (DPD) of radio frequency (RF) power amplifiers (PAs) using a modified decomposed vector rotation (DVR)-based behavioral model. To make the model hardware friendly, we first modify the model into a sub-decomposed format which significantly reduces the computational complexity in model extraction. We then reassemble the coefficients and propose a simple digital implementation structure for real-time signal processing in the transmit path. A new Dual-Direction COordinate Rotation DIgital Computer (DD-CORDIC) design is also proposed to simultaneously calculate both magnitude and $e^{j\theta_n}$ values to facilitate the model implementation. To validate hardware implementation, a wide-band signal is employed to evaluate the performance with a Doherty power amplifier. Experimental results show that the proposed approach can achieve comparable performance with much lower system complexity compared to that using the conventional approaches.**

***Index Terms* — behavioral model, CORDIC, digital predistortion, model extraction, power amplifier, radio frequency.**

I. Introduction

BENEFITING from the consistent scaling of CMOS technology, digital circuits are achieving excellent performance with low cost hardware and low power consumption. In the radio frequency (RF) area, digital predistortion (DPD) has been widely applied to linearization of RF power amplifiers (PAs) to achieve high power efficiency and simultaneously maintain linear signal amplification.

In the past decades, many advanced DPD models have been developed and the majority of the models used today are more or less modified from the Volterra series, such as memory polynomial (MP) [1]-[3], envelope-memory polynomial (EMP) [4], generalized memory polynomial (GMP) [5], dynamic deviation reduction (DDR) Volterra model [6], [7], and many others [8], [9]. Each of these models can achieve excellent performance in their specific application domains. However, because their basis functions are polynomial-based, the Volterra models have some inherent limitations. For instance, they are only best suitable for modeling continuously smooth and relatively weak nonlinear systems. With the continuous push towards wider bandwidth and higher efficiency, more and more advanced PA architectures, e.g., multi-way/multi-stage Doherty, coherent multiband and various switch-mode PAs, will emerge. In these systems, the PA can exhibit much stronger nonlinearities and the PA nonlinear behavior becomes far more complex. The existing Volterra models are facing significant challenges in modeling and linearizing these PAs.

Recently, a completely new behavioral model was proposed in [10]. This model was derived from a modified form of the canonical piecewise-linear function (CPWL) [11], [12] using a decomposed vector rotation (DVR) technique. In this model, the nonlinear operation is achieved by using the "absolute" value operation, which is completely different from the polynomials that are used in the Volterra models. Theoretical analysis has shown that this model is much more flexible in modeling RF PAs with non-Volterra-like behavior and experimental results confirmed that the new model can produce excellent performance with a relatively small number of coefficients compared to conventional models.

In [10], the author mainly focused on deriving the model structures and presenting system characterization methodologies. The model equation was only verified by software implementation, i.e., in MATLAB. Digital implementation of DPD is straightforward and the implementation cost of these digital units is usually relatively low compared to that of the high power RF amplifiers. In future systems, however, the cost and power consumption of digital circuits must be watched closely. It is because that, as operating bandwidths of wireless systems continue to increase, the nonlinear behavior of the PAs becomes more complicated, which leads that more complex DPD models are required to maintain the linearity and thus more complex numerical operations are needed that can increase the cost and power consumption. More importantly, in future networks, more and more small-cell base stations will be deployed [13], [14]. In these small cells, the output power of the PA becomes much lower and thus the relative power budget assigned for DPD must be much lower too. Therefore, in future DPD development, not only the performance is the concern, the

This work was funded by Science Foundation Ireland under Grant Numbers 12/IA/1267 and 13/RC/2077.

The authors are with the School of Electrical and Electronic Engineering, University College Dublin, Dublin 4, Ireland (e-mail: wenhui.cao@ucdconnect.ie; anding.zhu@ucd.ie).

hardware implementation of DPD, including the complexity of model coefficients extraction, must also be carefully considered.

In this paper, we introduce a power efficient and low-cost hardware implementation structure for implementing the DVR DPD model in digital circuits. First we modify the DVR model into a sub-decomposed format by alternatively selecting line segments in the model construction. This approach results in a significant reduction in computational complexity in model extraction. In the transmit path, we reassemble the coefficients and propose a very simple implementation structure that can dramatically reduce the complexity of signal generation in the predistorter. A new Dual-Direction COordinate Rotation DIgital Computer (DD-CORDIC) design is also proposed which simultaneously calculates both magnitude and $e^{j\theta_n}$ values to facilitate the model implementation.

This paper is organized as follows. Section II reviews the DVR model and current implementation complexity, followed by the derivation of the sub-decomposed DVR (SD-DVR) model and details of implementation methodologies in Section III. Section IV introduces the hardware structure of DD-CORDIC. Section V shows experimental results with a conclusion in Section VI.

II. Review of the DVR Model

The DVR model [10] can be expressed as

$$\begin{aligned}\tilde{u}(n)\big|_{DVR} = & \sum_{i=0}^{M}\tilde{a}_i\tilde{x}(n-i) \\ & +\sum_{k=1}^{K}\sum_{i=0}^{M}\tilde{c}_{ki,1}\left|\left|\tilde{x}(n-i)\right|-\beta_k\right|e^{j\theta(n-i)} \\ & +\sum_{k=1}^{K}\sum_{i=0}^{M}\tilde{c}_{ki,21}\left|\left|\tilde{x}(n-i)\right|-\beta_k\right|e^{j\theta(n-i)}\cdot\left|\tilde{x}(n)\right| \\ & +\cdots\end{aligned} \tag{1}$$

where $\tilde{x}(n)$ and $\tilde{u}(n)$ is the baseband input and output, respectively. The inner $|\cdot|$ returns the magnitude of $\tilde{x}(n)$, while the outer $|\cdot|$ is the normal real-valued "absolute" operation. θ_n represents the phase of $\tilde{x}(n)$. K denotes the number of partitions and β_k is the threshold that defines the boundary of the partitions. M represents the memory length. $\tilde{a}_i$ and $\tilde{c}_{ki,j}$ are the model coefficients. Because the nonlinear functions are composed in a piecewise manner, the DVR model does not have any restrictions on the shapes of the nonlinear curves. This model is therefore much more flexible than the Volterra models [10].

The generic block diagram of a DPD system is shown in Fig. 1: the input baseband signal $\tilde{x}(n)$ is predistorted by the predistorter block before being up-converted to RF frequency and sent to the PA. In order to extract the coefficients of the predistorter, a small fraction of the transmit signal is transferred back to baseband via a feedback loop. The model extraction unit compares the input and the output data and finds the coefficients for the predistorter.

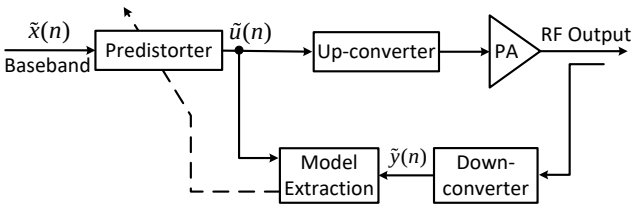

Fig. 1. Block diagram of a DPD system [20].

A. Complexity of Model Extraction

Since the model is linear-in-parameters, the general linear system identification algorithms, such as least squares (LS), can be applied in the model extraction. Let's rewrite the DVR in the matrix format as

$$U_{N\times 1} = X_{N\times Q}C_{Q\times 1} \tag{2}$$

where matrix X includes all the linear terms and various DVR interaction products constructed by using the input signal $\tilde{x}(n)$. The subscript N indicates the total number of input samples and Q represents the number of coefficients. Vector C contains the predistorter coefficients. The result vector U is the predistorted signal $\tilde{u}(n)$, going through the PA.

Two architectures are generally employed for model extraction: direct learning architecture (DLA) and indirect learning architecture (IDLA) [15], [16]. The DLA is usually used in closed-loop systems and compares the PA output with the original input directly. The IDLA estimates the post-inverse of PA first and then copies the coefficients to the pre-inverse block. By applying IDLA, the PA baseband output $\tilde{y}(n)$ is fed into the model, appearing as the model input instead of $\tilde{x}(n)$, to build the matrix Y. The predistorted output, $\tilde{u}(n)$, is specified as the model expected output. The coefficients can then be extracted by using the LS algorithm as following,

$$C_{Q\times 1} = [(Y^H)_{Q\times N}Y_{N\times Q}]^{-1}(Y^H)_{Q\times N}U_{N\times 1} \tag{3}$$

where $[\cdot]^H$ is conjugate transposition and $[\cdot]^{-1}$ indicates matrix inverse operation. The computational complexity of LS matrix computation can be broken down into several single matrix operations, listed in Table I, where only the complex multiplications are concerned, since the hardware cost of additions is negligible. The detailed analysis for the computational complexity of each matrix operation can be referred to [17]. When the size of the matrix is small, the computational complexity is out of problem. However, if the nonlinearity is quite strong, normally, around 8192 samples and 100 coefficients are required and thus approximately 165.6 million complex multiplications are needed, which leads to large computational complexity and occupies a large digital chip area.

TABLE I
COMPLEX MULTIPLICATIONS FOR EACH MATRIX OPERATION OF DVR

	Complex Multiplications
$S_1 = Y^H \times Y$	$N \times Q \times Q$
$S_2 = (S_1)^{-1}$	$Q \times Q \times Q$
$S_3 = S_2 \times Y^H$	$N \times Q \times Q$
$S_4 = S_3 \times U$	$N \times Q$
Total	$N \times Q^2 \times 2 + N \times Q + Q^3$

B. Complexity of Predistorter Implementation

In the transmit path, the main complexity is to implement the DPD model. For the DVR model, the main operation is to conduct the magnitude decomposition and phase restoration process. Let's take the 1st-order basis

$$\sum_{k=1}^{K}\sum_{i=0}^{M}\tilde{c}_{ki,1}\left|\left|\tilde{x}(n-i)\right|-\beta_k\right|e^{j\theta_{(n-i)}} \tag{4}$$

as an example. The first step is to generate the magnitude and phase from the complex data. This can normally be conducted by using a CORDIC.

The second step is to summarize all the decomposed components after multiplying with the coefficients. Since different delayed terms can reuse the same hardware structure, and the threshold index k is irrelevant to $e^{j\theta_n}$, the core unit to be implemented in the DVR model is

$$e^{j\theta_n}\sum_{k=1}^{K}\tilde{c}_k\left|\left|\tilde{x}(n)\right|-\beta_k\right|. \tag{5}$$

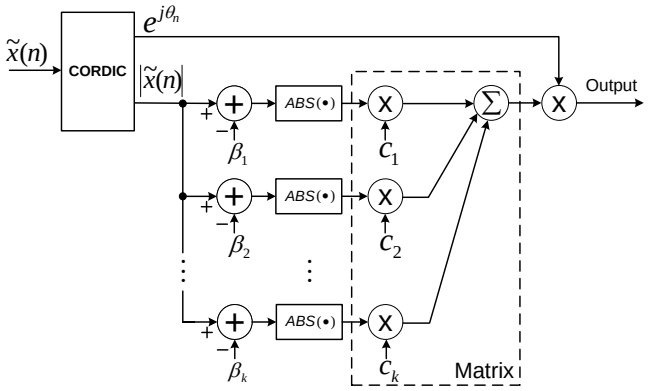

Fig. 2. Hardware implementation of nonlinear term (5) in DVR [10].

Implementing (5) is straightforward, as shown in Fig. 2, but it appears costly. $|\tilde{x}(n)|$ separately subtracts K threshold values β_k, and the outcomes then go through absolute operations. The output is the summation of K magnitude decompositions $\left|\left|\tilde{x}(n)\right|-\beta_k\right|$ multiplied with K complex coefficients $\tilde{c}_k$. If the summation is implemented in Xilinx FPGA, 2K DSP48 units (each DSP48 unit contains one multiplier and one adder) are required for both real and imaginary parts. With strong memory effects and nonlinearities involved in the model, the total number of DSP48s will increase by a factor of $M \times N_{term}$, where M represents memory delay and N_{term} designates the number of interaction products in the DVR model. The DSP48 unit is an important component in FPGA design, which requires dedicated hardware resource to implement, occupies a large die area and consumes most of the chip power. Using a large number of DSP48 units is not an economical choice.

III. SUB-DECOMPOSED DVR

In order to reduce the implementation complexity and reduce power consumption, in this work, we modify the DVR model to make it more hardware friendly without degrading the performance.

A. Model Modification

The magnitude decomposition in DVR can be represented as

$$\begin{aligned} G(\left|\tilde{x}(n)\right|,k) &= \left|\left|\tilde{x}(n)\right|-\beta_k\right|, \quad k=1,2,3,...,\mathrm{K} \\ &= \begin{cases} \left|\tilde{x}(n)\right|-\beta_k, & \left|\tilde{x}(n)\right| \ge \beta_k \\ \beta_k-\left|\tilde{x}(n)\right|, & \left|\tilde{x}(n)\right| < \beta_k \end{cases} \end{aligned} \tag{6}$$

where the threshold values are defined as $\beta_1 < \beta_2 < \cdots < \beta_k < \cdots < \beta_K$. Owning to the property of absolute value operation, it is easy to find in the geometrical construction that these pairs of line segments are symmetrical about the threshold value β_k. In other words, multiple pairs of line segments are symmetrically split at each threshold point.

In the literature [18], a simplicial canonical piecewise-linear (SCPWL) was proposed for nonlinear modeling. The SCPWL functions can be represented as

$$\begin{aligned} G_1(\left|\tilde{x}(n)\right|,k) &= 0.5\left(\beta_k-\left|\tilde{x}(n)\right|+\left|\beta_k-\left|\tilde{x}(n)\right|\right|\right), \quad k=1,2,3,...,\mathrm{K} \\ &= \begin{cases} 0 & \left|\tilde{x}(n)\right| \ge \beta_k \\ \beta_k-\left|\tilde{x}(n)\right| & \left|\tilde{x}(n)\right| < \beta_k \end{cases} \end{aligned}$$

or (7)

$$\begin{aligned} G_2(\left|\tilde{x}(n)\right|,k) &= 0.5\left(\left|\tilde{x}(n)\right|-\beta_k+\left|\left|\tilde{x}(n)\right|-\beta_k\right|\right), \quad k=1,2,3,...,\mathrm{K} \\ &= \begin{cases} \left|\tilde{x}(n)\right|-\beta_k & \left|\tilde{x}(n)\right| \ge \beta_k \\ 0 & \left|\tilde{x}(n)\right| < \beta_k \end{cases} \end{aligned}$$

(8)

Comparing SCPWL with DVR, we can find that SCPWL is simpler to implement because half of the terms are zero, but the disadvantage is that it only contains the half geometrical segments of DVR. For example, in (7), although $G_1(|\tilde{x}(n)|,k)$ depicts a two-segment piecewise-linear function, the right segment overlaps with the x-axis over $[\beta_k,+\infty)$, making no contribution to the model fitting. In the same manner, the function response of (8) expounds the other half segments of the DVR model, i.e., right-direction components. Only using the single-direction SCPWL cannot accurately model the nonlinearities in the entire input range.

To guarantee the performance, in this work, we propose to

employ the both-direction components, to cover the whole input range. We alternatively choose the left-direction segments on the odd-numbered positions from (7) and the right-direction segments on the even-numbered positions from (8) to create a new segment construction

$$F(|\tilde{x}(n)|,k)=\begin{cases}0.5(\beta_k-|\tilde{x}(n)|+|\beta_k-|\tilde{x}(n)||),k=1,3,5...2\rho\text{-}1\\0.5(|\tilde{x}(n)|-\beta_k+||\tilde{x}(n)|-\beta_k|),k=2,4,6...2\rho\end{cases} \tag{9}$$

where ρ is an integer index starting from 1. In (9), the first part depicts the line segments indexed by $k=2\rho-1$ and the second part describes the line segments on the even-numbered positions, defined by $k=2\rho$. Because the line segments in (7) and (8) can be considered as the sub-decompositions of magnitude decomposition in the DVR model, the new model is called as sub-decomposed DVR (SD-DVR):

$$\begin{aligned}\tilde{u}(n)|_{SD-DVR}=&\sum_{i=0}^{M}\tilde{a}_i\tilde{x}(n-i)\\&+\sum_{k=1}^{K}\sum_{i=0}^{M}\tilde{c}_{ki,1}F(|\tilde{x}(n-i)|,k)e^{j\theta(n-i)}\\&+\sum_{k=1}^{K}\sum_{i=0}^{M}\tilde{c}_{ki,21}F(|\tilde{x}(n-i)|,k)e^{j\theta(n-i)}\cdot|\tilde{x}(n)|\\&+\cdots\end{aligned} \tag{10}$$

B. Complexity Reduction of Model Extraction

Interestingly, the computational complexity of model extraction for the SD-DVR is dramatically reduced. As discussed in Section II, to extract model coefficients, we first need to gather a set of data samples and then build data vectors and matrixes for the LS estimation. The key matrix to be built is the matrix Y in (3) where each row includes the linear and nonlinear model terms. For instance, the first row can include $[\tilde{y}(n), \tilde{y}(n-1), \dots, |\,|\tilde{y}(n)\,|-\beta_1|e^{j\theta_n}, \dots]$, and the second row has the form of $[\tilde{y}(n-1), \tilde{y}(n-2), \dots, |\,|\tilde{y}(n-1)\,|-\beta_1|e^{j\theta_{(n-1)}}, \dots]$, and so on. As explained earlier, the nonlinearity composition via DVR is the superposition of multiple pairs of line segments symmetrically split at each threshold point. Each pair of segments splits the input range into two parts, and which part of the segments is chosen depends on the magnitude of the input signal sample comparing with the threshold value, namely, the right-direction segment is chosen if the input signal is greater than the threshold value otherwise the left-direction segment is chosen. Nevertheless, all the nonlinear elements in the matrix Y have values, e.g., $(|\tilde{y}(n)\,|-\beta_1)e^{j\theta_n}$, $(\beta_3-|\tilde{y}(n)\,|)e^{j\theta_n}$. If we use $\times$ to indicate a non-zero value, the matrix Y in the DVR model can be written as

$$Y_{DVR}=\begin{bmatrix}\times&\times&\times&\cdots&\times&\times\\\times&\times&\times&\cdots&\times&\times\\\vdots&\vdots&\vdots&\vdots&\vdots&\vdots\\\times&\times&\times&\cdots&\times&\times\end{bmatrix}. \tag{11}$$

In SD-DVR, we alternatively select one segment from each pair in original DVR, "flattening" the other segment. Therefore, for the same input samples, SD-DVR only has approximately half valid segments in comparison with that of DVR. This leads that, approximately, half of the nonlinear elements in the matrix Y are zeros,

$$Y_{SD-DVR}=\begin{bmatrix}\times&\times&\times&0&\times&\cdots&\times&0\\\times&\times&0&\times&\times&\cdots&\times&0\\\vdots&\vdots&\vdots&\vdots&\vdots&\vdots&\vdots&\vdots\\\times&\times&0&\times&0&\cdots&0&\times\end{bmatrix} \tag{12}$$

These zeros will directly affect the matrix multiplication operations in S_1 and S_3. Because there is an approximately 50% possibility for each element in matrix Y to be zero, only 25% complex multiplications are finally required for conducting matrix multiplication $S_1=Y^H\times Y$, comparing to that in DVR. As listed in Table II, S_3 matrix multiplication is also directly affected by Y^H, where only 50% computational complexity is required. Even though operations S_1 and S_3 are just part of the LS estimation, those two large matrix operations are the main factors of computational complexity. Reducing their complexities will directly affect the overall system complexity, which can be seen in the experimental result section in more detail.

TABLE II
COMPLEX MULTIPLICATIONS FOR EACH MATRIX OPERATION OF SD-DVR

	Complex Multiplications
$S_1=Y^H\times Y$	$N\times Q\times Q\times 1/4$
$S_2=(S_1)^{-1}$	$Q\times Q\times Q$
$S_3=S_2\times Y^H$	$N\times Q\times Q\times 1/2$
$S_4=S_3\times U$	$N\times Q$
Total	$N\times Q^2\times 3/4+N\times Q+Q^3$

C. Complexity Reduction of Predistorter Implementation

When it comes to the hardware implementation of the predistorter, the core component of SD-DVR is

$$e^{j\theta_n}\sum_{k=1}^{K}\tilde{c}_kF(|\tilde{x}(n)|,k) \tag{13}$$

where $F(|\tilde{x}(n)|,k)$ is the nonlinear function defined by (9). To simplify the following derivation, the memory and high order terms are omitted here. Similar to that discussed in Section II, direct implementation of (13) will require a large number of dedicated DSP units. In this section, we propose to implement (13) in a much simpler way.

In the predistorter, the model coefficients $\tilde{c}_k$ are known after model extraction and the partition thresholds β_k are also pre-determined, the summation $\sum_{k=1}^{K}\tilde{c}_kF(|\tilde{x}(n)|,k)$ result in (13) turns out only depending on the magnitude value of the input signal, i.e., $|\tilde{x}(n)|$. This leads that a low-cost implementation strategy can be applied, similar to that discussed in [19]-[21]. The idea is to merge all the valid

coefficients together before multiplying with $|\tilde{x}(n)|$, as explained below.

Assuming $|\tilde{x}(n)|$ is greater than a particular threshold value β_{k-1} and less than the upper adjacent threshold β_k, here we take the left-direction segments, the function with $k = 1, 3, \ldots$ in (9), as an example. Only the coefficients with index greater than k multiply with non-zero values while the rest of coefficients can be ignored because the magnitude decompositions are zeros. Thus the summation of left-direction segments is $-|\tilde{x}(n)| \cdot \sum_{i=\lceil (k-1)/2 \rceil}^{\lfloor (K-1)/2 \rfloor} \tilde{c}_{2i+1} + \sum_{i=\lceil (k-1)/2 \rceil}^{\lfloor (K-1)/2 \rfloor} \tilde{c}_{2i+1}\, \beta_{2i+1}$. Applying the same rule on the right-direction segments, the summation is $|\tilde{x}(n)| \cdot \sum_{i=1}^{\lfloor (k-1)/2 \rfloor} \tilde{c}_{2i} - \sum_{i=1}^{\lfloor (k-1)/2 \rfloor} \tilde{c}_{2i}\, \beta_{2i}$. By merging them together, (13) can be re-written as

$$
\begin{aligned}
& e^{j\theta_n} \sum_{k=1}^{K} \tilde{c}_k F(|\tilde{x}(n)|, k) \\
& = e^{j\theta_n} \left\{ |\tilde{x}(n)| \left[\sum_{i=1}^{\lfloor (k-1)/2 \rfloor} \tilde{c}_{2i} - \sum_{i=\lceil (k-1)/2 \rceil}^{\lfloor (K-1)/2 \rfloor} \tilde{c}_{2i+1} \right] + \left[- \sum_{i=1}^{\lfloor (k-1)/2 \rfloor} \tilde{c}_{2i} \beta_{2i} + \sum_{i=\lceil (k-1)/2 \rceil}^{\lfloor (K-1)/2 \rfloor} \tilde{c}_{2i+1} \beta_{2i+1} \right] \right\} \\
& = e^{j\theta_n} \left\{ |\tilde{x}(n)| A_k + B_k \right\}, \qquad \beta_k > |\tilde{x}(n)| \geq \beta_{k-1}
\end{aligned}
\tag{14}
$$

where $\lfloor \cdot \rfloor$ rounds the elements to the nearest integers towards minus infinity and $\lceil \cdot \rceil$ rounds the elements to the nearest integers towards infinity. A_k and B_k represents gain and offset, respectively. The index k is an integer between 1 and $(K+1)$, categorizing $K+1$ coefficient groups for different magnitude zones.

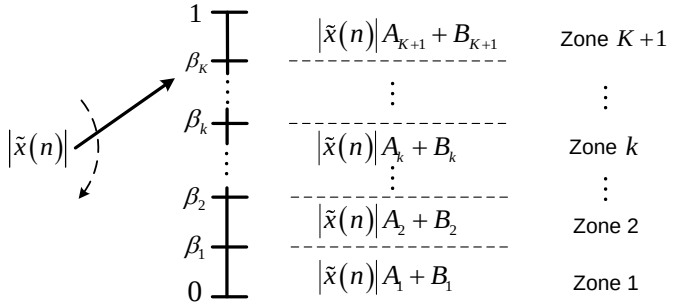

Fig. 3. The summation demonstration in (14) for SD-DVR.

In this arrangement, the implementation complexity and power consumption can be significantly reduced compared to that using the direct implementation. In the direct implementation, every single input sample is compared with the thresholds first to obtain the magnitude difference and then multiplied with each coefficient and finally summed together. In the proposed approach, after compared with the thresholds, the input samples are directed into separate zones, defined by the threshold values, depending on their magnitude values, as shown in Fig. 3. For instance, for input samples, $0.6 + 0.1j$ and 0.1-0.04j, $0.6 + 0.1j$ can be directed into zone 4 while 0.1-0.04j into zone 1. To obtain the output, these samples are multiplied with the "merged" coefficients plus offsets in the corresponding zones, i.e., $|0.6 + 0.1j| \times A_4 + B_4$ and $|0.1 - 0.04j| \times A_1 + B_1$, respectively.

The hardware implementation is depicted in Fig. 4, where the sets of A_k and B_k are built into the look-up table (LUT) and the signal magnitude $|\tilde{x}(n)|$ is used as the index to select which set of coefficients is used for the output calculation. In this implementation, the nonlinear signal process only requires one complex multiplication and one complex addition (two DSP48 units). Compared to the hardware cost of $2K$ DSP48s in the traditional implementation in Fig. 2, the new design substantially cuts down the computational complexity and hardware cost. The detailed improvement in the hardware efficiency is shown in the result section.

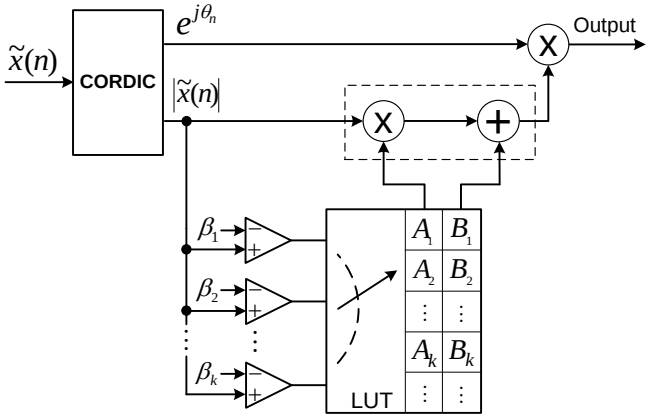

Fig. 4. The hardware implementation of term (14) for SD-DVR.

IV. Dual-Direction CORDIC

The overall model structure has been addressed. The next task is to obtain the magnitude $|\tilde{x}(n)|$ and the phase restoration information $e^{j\theta_n}$ to feed into the model. The general procedure is: (i) produce $|\tilde{x}(n)|$ and angle θ_n from $\tilde{x}(n)$ by using the CORDIC algorithm [22], [23]; (ii) obtain $e^{j\theta_n}$ according to the value of θ_n: $e^{j\theta} = cos\,\theta + j\,sin\,\theta$; (iii) feed$|\tilde{x}(n)|$ and $e^{j\theta_n}$ into the DVR model to finish the following computation. This procedure is straightforward, but it involves multiple calculations.

If we carefully re-look the equation, we can find that we actually do not need to calculate the phase θ_n. It is because what we need to know is $e^{j\theta_n}$, which is a unit complex number. Rewriting $\tilde{x}(n)$ as $|\tilde{x}(n)|e^{j\theta_n}$, we will find that the only difference between $\tilde{x}(n)$ and $e^{j\theta_n}$ is their magnitude value: $|\tilde{x}(n)|$ for $\tilde{x}(n)$ and one for $e^{j\theta_n}$, but the two complex numbers share the same phase.

The basic idea of CORDIC is to iteratively rotate the vector to a new vector ending up with zero phase and then the real part of the new vector is the magnitude, as the imaginary part is zero. For instance, as shown in Fig. 5 (a), to generate the magnitude, the vector $\tilde{x}$ is rotated clockwise by phase θ to the Re-axis, so that the magnitude of $\tilde{x}$ can be obtained from reading the real part of the vector A. This way can be reversely

applied to $e^{j\theta}$. The trick here is to add a unit vector whose coordinate is $E(1,0)$, shown in Fig. 5 (b). If we rotate it counterclockwise by the same phase with constant radius 1, we can obtain the vector B which is perfectly coinciding with the vector $\tilde{x}$. The real and imaginary part of $e^{j\theta}$ can then be obtained by reading from the vector B, which is $cos\theta$ and $j\,sin\,\theta$, respectively. The clockwise and counterclockwise rotations can be coordinately operated by using one set of shared digital logic as described below.

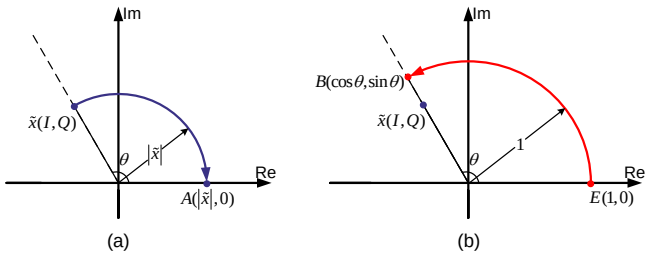

Fig. 5. Vector rotation demonstration: (a) clockwise rotation for magnitude $|\tilde{x}|$, (b) counterclockwise rotation for $e^{j\theta}$.

A. *Magnitude $|\tilde{x}|$ Generation*

Firstly, a detailed demonstration of calculating the magnitude by using CORDIC is introduced. For example, for a complex number $\tilde{x} = I + jQ$, to obtain its magnitude, the first step is to make sure its phase is located in the range of $[-90^{\circ},\ 90^{\circ}]$. This can be conducted by checking the sign of Q. If Q is positive, rotate the vector clockwise by 90 degrees, otherwise counterclockwise by 90 degrees.

The following operation is to rotate the vector step by step, approaching the Re-axis. Let's define a complex number as "$1 + jR$", whose phase can be represented as $\tan^{-1}(R)$. Therefore, adding a phase of $\tan^{-1}(R)$ to the current vector (counterclockwise rotation), it is equivalent to multiplying with "$1 + jR$", otherwise multiplying with "$1 - jR$". The sign of $\pm$ indicates the rotation direction. Importantly, the value of R decreases with powers of two after each rotation [22], starting with $2^0 = 1$ and then $2^{-1}, 2^{-2}, 2^{-3}, 2^{-4}$ and so on. The corresponding rotation phase $\tan^{-1}(R)$ approaches zero after a number of iterations.

The iterative rotation can be described as

$$\begin{bmatrix} I_{i+1} \\ Q_{i+1} \end{bmatrix} = \delta_i \begin{bmatrix} 1 & -\sigma_i 2^{-i} \\ +\sigma_i 2^{-i} & 1 \end{bmatrix} \begin{bmatrix} I_i \\ Q_i \end{bmatrix} \qquad (15)$$

where I_i and Q_i represents the real and imaginary part of the complex number, respectively, and i designates the i^{th} rotation. σ_i is defined as the value of -1 or 1, which decides the direction of rotation. The gain of each rotation is $\delta_i = 1/\sqrt{1 + 2^{-2i}}$. The ultimately accumulated gain is $\delta_{total} = \prod_i \delta_i = \prod_i (1/\sqrt{1 + 2^{-2i}})$, which can be compensated together. In digital logic, multiplying with powers of two can be easily conducted by using bit shifts instead of actual multipliers. CORDIC therefore significantly reduces the implementation cost.

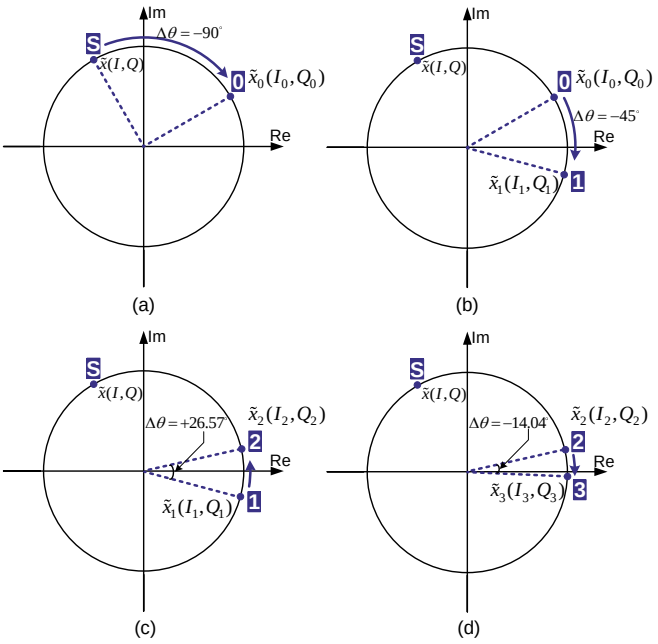

Fig. 6. CORDIC rotation demonstration: (a) relocate vector to the range of $[-90^{\circ},\ 90^{\circ}]$, (b) 0^{th} iteration, (c) 1^{st} iteration and (d) 2^{nd} iteration.

To further demonstrate how the CORDIC works, a step-by-step illustration is given in Fig. 6. The vector $\tilde{x}(I,Q)$ at the starting point is rotated clockwise by 90 degrees ($Q > 0$) to $\tilde{x}_0$, relocated in the range of $[-90^{\circ},\ 90^{\circ}]$. To obtain the coordinate of $\tilde{x}_0$, exchange I and Q, and flip the sign of the imaginary part, i.e., $I_0 = Q, Q_0 = -I$. If the vector is rotated counterclockwise by 90 degrees, then exchange I and Q, and flip the sign of the real part.

TABLE III
ITERATIVE ROTATION DATA LIST FOR CORDIC

Coordinate	Position (i)	Direction (σ)	R	$ant^{-1}(R)$
$\tilde{x}(I,Q)$	Start			
$\tilde{x}_0(I_0,Q_0)$	0	-1	2^0	45°
$\tilde{x}_1(I_1,Q_1)$	1	+1	2^{-1}	26.57°
$\tilde{x}_2(I_2,Q_2)$	2	-1	2^{-2}	14.04°
$\tilde{x}_3(I_3,Q_3)$	3	+1	2^{-3}	7.125°
….	…	…	…	…

The following rotation details are listed in Table III. We take $\tilde{x}_1$ as an example to illustrate the iterative process in Fig. 6 (c). Due to the fact that $Q_1 < 0$, the next rotation is to add the phase of $tan^{-1}(2^{-1}) = 26.57^{\circ}$ with the current vector $\tilde{x}_1$. We need to multiply "$1 + j \times 2^{-1}$" with "$I_1 + jQ_1$" to obtain new vector $\tilde{x}_2$. According to (15), the real value of $\tilde{x}_2$ is $I_2 = I_1 - 2^{-1} \times Q_1$, whose operation is shifting Q_1 to right by 1 bit, then subtracted from I_1, without considering error compensation δ_1. The same binary arithmetic can be performed on the imaginary part: $Q_2 = 2^{-1} \times I_1 + Q_1$. In the following 2^{nd} rotation,

because of $Q_2 > 0$, the next rotation is to subtract the phase of $tan^{-1}(2^{-2}) = 14.04^{\circ}$ from the current vector $\tilde{x}_2$. The rest procedure can be done in the same manner. After a certain amount of rotations, the value on the Re-axis can be approximated as the magnitude of the vector since the phase approaches zero.

B. *Phase Restoration Information $e^{j\theta}$ Generation*

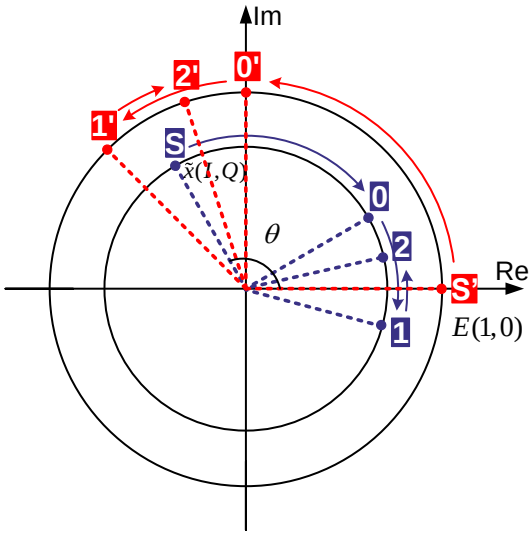

Fig. 7. Demonstration of dual-direction CORDIC operations.

As discussed earlier, the value of $e^{j\theta}$ can be obtained by concurrently rotating a vector E by the exact same phases as the vector $\tilde{x}$ has done, but in an "opposite" direction. To illustrate this process, we add the other trace into the CORDIC trace of rotating $\tilde{x}$ and label the iteration steps with the "prime" sign, as shown in Fig. 7. When "S" moves to "0", "S'" moves to "0'". With the aid of the iteration trace, E (1, 0) can be rotated all the way back to get $e^{j\theta}$. It shows that when the vector $\tilde{x}$ reaches the Re-axis, vector E ends at the position indicating the same phase θ as the original vector $\tilde{x}$, straightforward offering the value of $e^{j\theta} = cos\,\theta + j\,sin\,\theta$.

Based on the above analysis, the iterative rotation for $e^{j\theta}$ can be described as

$$\begin{bmatrix} I_{i+1} \\ Q_{i+1} \end{bmatrix} = \delta_i \begin{bmatrix} 1 & +\sigma_i 2^{-i} \\ -\sigma_i 2^{-i} & 1 \end{bmatrix} \begin{bmatrix} I_i \\ Q_i \end{bmatrix} \tag{16}$$

where the difference between (15) and (16) is the sign of $\pm\sigma$, which serves as the rotation direction. Therefore, the dual-direction iterative matrix can be built as

$$\begin{bmatrix} I_{i+1}^{(A)} \\ Q_{i+1}^{(A)} \\ I_{i+1}^{(B)} \\ Q_{i+1}^{(B)} \end{bmatrix} = \delta_i \begin{bmatrix} 1 & -\sigma_i 2^{-i} & 0 & 0 \\ +\sigma_i 2^{-i} & 1 & 0 & 0 \\ 0 & 0 & 1 & +\sigma_i 2^{-i} \\ 0 & 0 & -\sigma_i 2^{-i} & 1 \end{bmatrix} \begin{bmatrix} I_i^{(A)} \\ Q_i^{(A)} \\ I_i^{(B)} \\ Q_i^{(B)} \end{bmatrix} \tag{17}$$

where the first two rows (process A) define the calculation of magnitude $|\tilde{x}|$, starting from vector $\tilde{x}$, and the last two rows (process B) state the process of $e^{j\theta}$, rotating from vector E.

After several iterations, e.g., N, the $Q_N^{(A)}$ will be close to 0 and then the rotation will stop. The magnitude $|\tilde{x}|$ and $e^{j\theta}$ can then be obtained,

$$\begin{bmatrix} |\tilde{x}| \\ 0 \\ \cos\theta \\ \sin\theta \end{bmatrix} \approx \begin{bmatrix} I_N^{(A)} \\ Q_N^{(A)} \\ I_N^{(B)} \\ Q_N^{(B)} \end{bmatrix}. \tag{18}$$

Two rotation traces can be implemented by using the same digital logic, as the rotation mechanisms are exactly the same, except that they add opposite phases in each rotation. Therefore the proposed CORDIC is named "Dual-Direction" CORDIC (DD-CORDIC) in our paper.

C. *Time Multiplexing Dual-Direction CORDIC Design*

Time-division multiplexing has been widely used in digital design to reduce hardware cost with increased clock rates. Facilitated by the time-division multiplexer, process A and B in (17) can be implemented by using only one dual-direction rotation structure depicted in Fig. 8. At the sampling frequency, the data stream $\tilde{x}(n)$ and E are fed into the digital block, and the multiplexer cooperating with CS alternately captures vectors $\tilde{x}(n)$ or E at twice speed of the sampling rate.

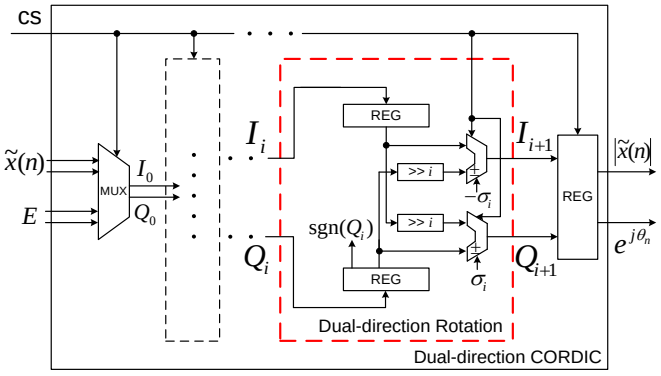

Fig. 8. The DD-CORDIC process with the multiplexing technique.

Since the DD-CORDIC includes many iterative rotation units which are pipelined in a sequence, the structure of one dual-direction rotation unit is highlighted by the dashed rectangle. The rotation phase of each rotation unit is pre-set and the imaginary value of vector $\tilde{x}(n)$ decides the rotation direction. The rotation process is accomplished with shifts and adders. In the next clock cycle, CS signal flips the direction of rotation, operating an opposite rotation upon vector E. The alternate data flow of $\tilde{x}(n)$ and E within DD-CORDIC is double the speed of the input sampling rate. To explain the variation of data flow, a time sequence for the DD-CORDIC is

provided in Fig. 9. In the end, the outputs $|\tilde{x}(n)|$ and $e^{j\theta_n}$ are down-sampled to the original sampling speed, in order to maintain a consistent communication rate between different blocks.

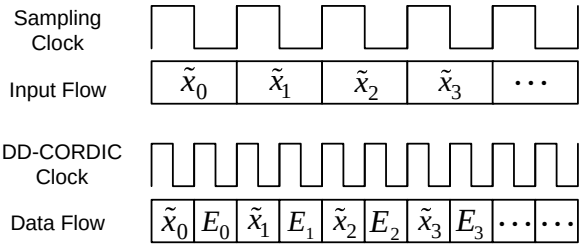

Fig. 9. Time sequence of the DD-CORDIC.

Note that only shifters and adders are involved in the dual-direction rotation unit, while compensation gain δ_{total} is omitted here. This is because, normally, the gain δ_{total} may not need to be compensated for or be compensated for during other procedure. In our test, we multiply $\delta_{total}$with coefficients $\tilde{c}_{ki}$, compensating the gain error in the offline characterization process.

V. Experimental Results

To validate the proposed approach, various experimental tests were conducted. Regarding model extraction, the computational complexity for the proposed SD-DVR model was discussed in comparison with that of the DVR model. After extracting the coefficients, the DPD block was implemented on FPGA and linearization performance was measured by normalized mean square errors (NMSE) and adjacent channel power ratio (ACPR). Comparisons were also given in terms of hardware resource utilization of the two models.

The DPD test platform employed was the same as that used in [10], which includes a PC, a baseband FPGA board, an RF board and a PA, as shown in Fig. 10. The baseband in-phase and quadrature (I/Q) digital signal source was generated by using either software in MATLAB or FPGA hardware board. The baseband signal was then sent to the RF board for modulation and up-conversion to RF frequency by the transmit (Tx) chain and finally sent to the PA. To reduce peak-to-average power ratio (PAPR), a crest factor reduction block (CFR) was applied on the input signal. For model extraction, a fraction of the output signal was down-converted to baseband and captured by the receiver (Rx) chain through the feedback path and sent back to the PC. The time alignment and model extraction were conducted off-line in MATLAB.

The PA under test was an in-house designed broadband Doherty power amplifier (DPA) operating at 2.14 GHz and excited by a 60 MHz 12 carriers UMTS signal with 6.5 dB peak-to-average power ratio (PAPR) and 32.36 dBm average output power. Approximately, 16,000 I/Q samples were saved at a sampling rate of 368.64 MSPS. Recorded complex input and output samples were time aligned and normalized before training the model. The FPGA board employed for DPD hardware implementation was Virtex-7 XC7VX485T, whose operating clock frequency was designed as 260 MHz (and 520 MHz for multiplexer's clock of DD-CORDIC). The same nonlinear terms, the 2nd-order type-1, 2 and 3 in [10], of DVR and SD-DVR were selected from (1) and (10), respectively. K was set as 7 and $M = 3$, resulting 74 coefficients in total.

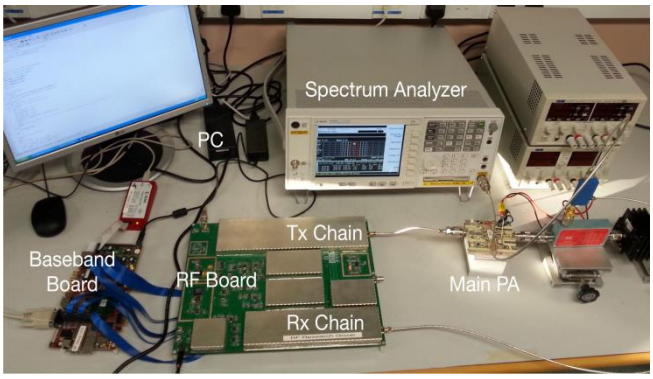

Fig. 10. DPD test bench.

A. Model Extraction

The LS algorithm was chosen for model extraction and 8192 training samples were used. As analyzed earlier, the LS process can be split into S_1, S_2, S_3 and S_4 operations. Multiplications S_2 and S_4 were considered as the same matrix operations for both DVR and SD-DVR. Because the elements in $\boldsymbol{Y}$ matrixes in the two models have different values, the computational complexity of S_1 and S_3 are different.

The matrix Y of SD-DVR contained more or less half zeros and half valid values. On the contrary, the elements in matrix Y of DVR appeared to be full valid values. To compare the complexity, we only took the nonlinear terms into account first, which involved 70 coefficients. The number of required complex multiplications and the complexity reductions of each operation between two models are summarized in Table IV. For S_1, each element in both $Y$and Y^H had an approximately 50% chance of being zero, leading to 74.81% complexity reduction. Step S_3 was only affected by matrix Y^H, reducing computation by 49.96%. In total, approximately 62% saving can be made in the complex multiplications. Including all the terms, i.e., the complete 74 coefficients, the complexity comparison is shown in Table V. Even though the linear terms use the same resources in both models, the total resource saving of the complete model extraction is still around 60%, which is significant.

TABLE IV
Complexity Reduction of Nonlinear Terms

	Complex Multiplications		Reduction
	DVR	SD-DVR	
$S_1 = Y^H \times Y$	40,140,800	10,113,295	74.81%
$S_2 = (S_1)^{-1}$	343,000	343,000	0
$S_3 = S_2 \times Y^H$	40,140,800	20,086,990	49.96%
$S_4 = S_3 \times U$	573,440	573,440	0
Total	81,230,808	31,116,725	61.69%

TABLE V
COMPLEXITY REDUCTION OF THE COMPLETE MODEL

	Complex Multiplications		Reduction
	DVR	SD-DVR	
$S_1 = Y^H \times Y$	44,859,392	12,540,023	72.05%
$S_2 = (S_1)^{-1}$	405,224	405,224	0
$S_3 = S_2 \times Y^H$	44,859,392	23,659,650	47.26%
$S_4 = S_3 \times U$	606,208	606,208	0
Total	90,730,216	36,613,163	59.65%

B. *Comparisons of DPD Performances*

The obtained coefficients were applied in the DPD hardware implementation. For comparison, we implemented DVR and SD-DVR differently. For DVR, we used standard CORDIC, where we calculated both magnitude and phase θ_n first and then obtained $e^{j\theta_n}$ according to the phase θ_n. In SD-DVR, we applied DD-CORDIC to obtain magnitude and $e^{j\theta_n}$ simultaneously. Given the overall structure of DPD model, the straightforward strategy in Fig. 2 was deployed for DVR, while SD-DVR used the simplified structure (14), shown in Fig. 4, to save hardware resources.

For the purpose of fair comparison, some standards of hardware design were kept the same for both approaches. For example, each I/Q signal of both input and output was kept as 32-bit data, in which the least significant 16 bits indicated the real part and the most significant 16 bits designated the imaginary part. The bit precision in the intermediate process, such as adders, multiplexers and configuration of DSP48 units, remained the same for both approaches. To satisfy the desired accuracy, the rotation number of CORDIC and DD-CORDIC was 9.

To verify the linearization performance, we not only compared the performances between the two models, but also the performances between the software/MATLAB DPD implementations and the corresponding hardware/FPGA ones. As expected, after cascading DPD and PA chain, the PA nonlinearities were almost completely removed by both hardware and software predistorters. It can be seen from Table VI that the ACPRs ($\pm$5 MHz) of the 60 MHz signal in all cases were suppressed to -51 dBc from -28 dBc and NMSEs were reduced to around -40 dB. The same DPD performance was achieved in hardware as that in software, which demonstrated high accuracy of the hardware DPD implementation.

TABLE VI
LINEARIZATION PERFORMANCE COMPARISON

DPD Calibration	NMSE (dB)	ACPR (dBc) ($\pm$5 MHz)
Without DPD	-15.50	-28.81/-25.51
DVR MATLAB	-40.20	-53.88/-51.34
DVR FPGA	-40.21	-53.76/-51.26
SD-DVR MATLAB	-40.48	-53.97/-51.74
SD-DVR FPGA	-40.08	-53.58/-51.35

Moreover, the spectrum comparison between DVR and SD-DVR (hardware only) is shown in Fig. 11. To illustrate performance, AM/AM and AM/PM characteristics after linearization via hardware DPDs are shown in Fig. 12. Those results validated that both hardware implementations achieved the same linearization level.

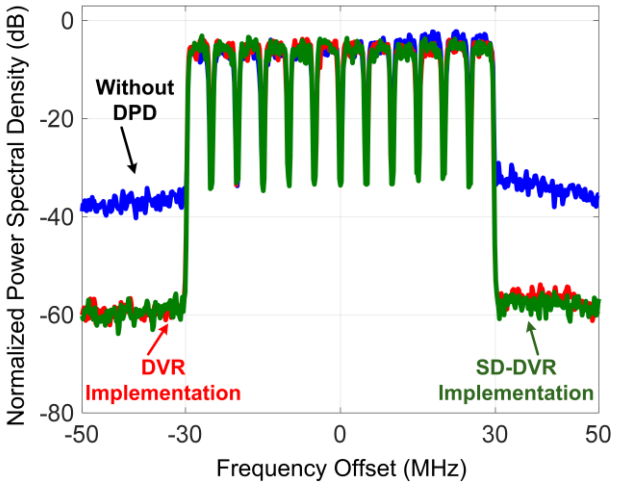

Fig. 11. Output spectra comparison for 60 MHz, 12-carrier UMTS signal.

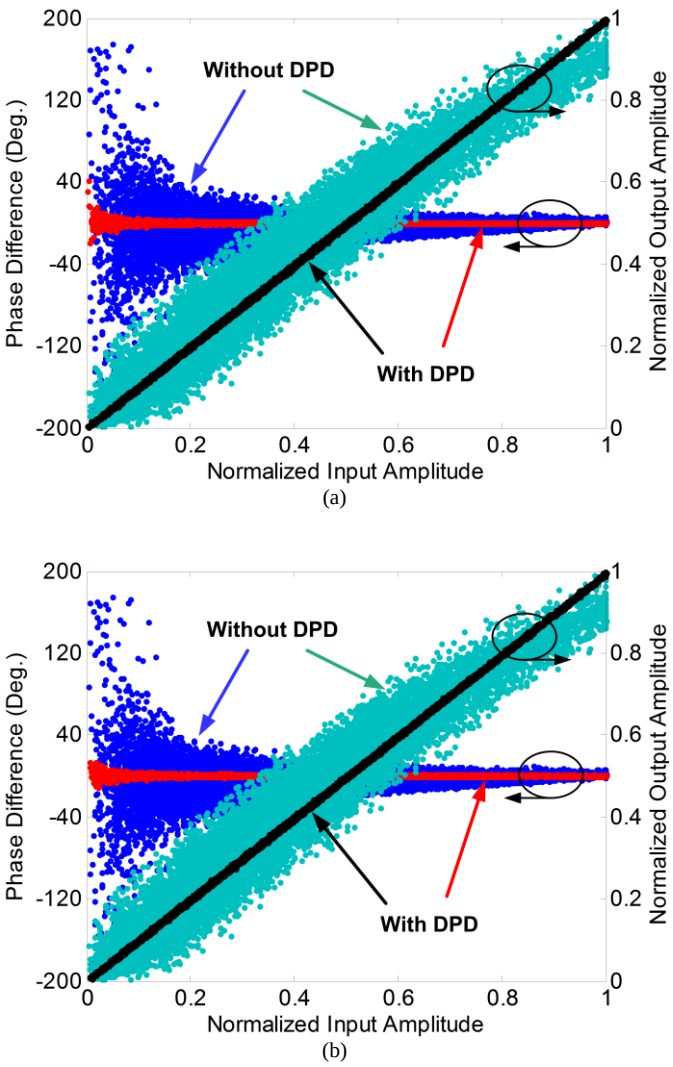

Fig. 12. AM/AM and AM/PM plots for linearization performances comparison with and without hardware DPDs: (a) DVR (b) SD-DVR.

C. Comparisons of DPD Resource Utilizations

The hardware implementation process was conducted in 5 steps: the first step was to calculate the magnitude and $e^{j\theta_n}$; linear term, i.e., $\sum_{i=0}^{M} a_i \tilde{x}(n-i)$, was regarded as the second step which was identical for both models; the third step was to implement the summation of nonlinear terms, e.g., $\sum_{k=1}^{K} \tilde{c}_k G(|\tilde{x}(n-i)|, k)$, $\sum_{k=1}^{K} \tilde{c}_k F(|\tilde{x}(n-i)|, k)$; the accomplishment of phase restoration or various interaction of present and past samples, memory delays and final nonlinear summation were carried out in step 4; linear and nonlinear terms were summed up in the last step. The other resources were used to synchronize the signal.

The resource utilizations of DVR and SD-DVR are listed in Table VII and VIII, respectively. Overall, 60% Flip-flops and 34% Slice LUTs were reduced during step 1 after the DD-CORDIC was applied. Importantly, the low-cost structure in step 3 significantly saved 85.7% DSP48 units, 98.11% Flip-flops and 95.31% Slice LUTs, comparing with the traditional implementation. In total, the new approach eliminated 28327 (63.5%) Flip-flops, 6478 (28.1%) Slice LUTs and 120 (85.7%) DSP48 units. The large saving of DSP48 in FPGA certainly makes the proposed approach more competitive in industry. The DD-CORDIC is also valuable for the application of DVR-like models, involving both magnitude and phase restoration $e^{j\theta_n}$.

TABLE VII
TRADITIONAL IMPLEMENTATION FOR DVR

	Block resource	FF	Slice LUT	DSP48
Step 1	CORDIC e.g.$\|\tilde{x}(n)\|e^{j\theta_n}$	2153	1628	
Step 2	Linear term	3970	4314	
Step 3	$\sum \tilde{c}_k G(\|\tilde{x}(n-i)\|, k)$	27556	6135	140
Step 4	Nonlinear summation	10617	10723	
Step 5	Sum up	32	32	
Others	Synchronization	256	192	
	Total	44584	23024	140

TABLE VIII
LOW-COST IMPLEMENTATION FOR SD-DVR

	Block resource	FF	Slice LUT	DSP48
Step 1	DD-CORDIC e.g.$\|\tilde{x}(n)\|e^{j\theta_n}$	860	1061	
Step 2	Linear term	3970	4314	
Step 3	$\sum \tilde{c}_k F(\|\tilde{x}(n-i)\|, k)$	521	288	20
Step 4	Nonlinear summation	10617	10723	
Step 5	Sum up	32	32	
Others	Synchronization	257	128	
	Total	16257	16546	20

VI. CONCLUSION

This paper has proposed the SD-DVR model for PA linearization, whose computational complexity of the model extraction has been significantly simplified. Moreover, we have introduced an efficient hardware structure for the predistorter to further reduce implementation complexity and power dissipation, facilitating future small cell applications. A dual-direction CORDIC has also been proposed to simultaneously calculate both magnitude and $e^{j\theta_n}$ values. The experimental results and hardware resource utilization have been provided to validate the functionality of the new model and its implementation methodologies.

REFERENCES

[1] L. Ding, G. T. Zhou, D. R. Morgan, Z. Ma, J. S. Kenney, J. Kim, and C. R. Giardina, "A robust digital baseband predistorter constructed using memory polynomials," *IEEE Trans. Commun.*, vol. 52, no. 1, pp. 159–165, Jan. 2004.

[2] J. Kim and K. Konstantinou, "Digital predistortion of wideband signals based on power amplifier model with memory," *Electron. Lett.*, vol. 37, no. 23, pp. 1417–1418, Nov. 2001.

[3] B. Ai, Z. Zhong, T. Jiang, and B. Li, "Novel pre-distortion of power amplifier with proposed fractional order memory polynomial," *Global Telecommunications Conference*, pp. 1-6, Dec. 2011.

[4] O. Hammi, F. M. Ghannouchi, and B. Vassilakis, "A compact envelope-memory polynomial for RF transmitters modeling with application to baseband and RF-digital predistortion," *IEEE Microw. Compon. Lett.*, vol. 18, no. 5, pp. 359–361, May 2008.

[5] D. R. Morgan, Z. Ma, J. Kim, M. G. Zierdt, and J. Pastalan, " A generalized memory polynomial model for digital predistortion of RF power amplifiers," *IEEE Trans. Signal Process.* vol. 54, no. 10, pp. 3852-3860, Oct. 2006.

[6] A. Zhu, J. C. Pedro, and T. J. Brazil, "Dynamic deviation reduction-based Volterra behavioral modeling of RF power amplifiers," *IEEE Trans. Microw. Theory Techn.*, vol. 54, no. 12, pp. 4323–4332, Dec. 2006.

[7] A. Zhu, P. J. Drexler, J. J. Yan, T. J. Brazil, D. F. Kinball, and P. M. Asbeck, "Open-loop digital predistorter for RF power amplifiers using dynamic deviation reduction-based Volterra series," *IEEE Trans. Microw. Theory Techn.*, vol. 56, no. 7, pp. 1524-1534, Jul. 2008.

[8] J. Wood, *Behavioral Modeling and Linearization of RF Power Amplifiers*. Norwood, MA: Artech House, 2014.

[9] F.-L. Luo, *Digital Front-End in Wireless Communications and Broadcasting: Circuits and Signal Processing*. Cambridge, U.K.: Cambridge Univ. Press, 2011.

[10] A. Zhu, "Decomposed vector rotation-based behavioral modeling for digital predistortion of RF power amplifiers," *IEEE Trans. Microw. Theory Techn.*, vol. 63, no. 02, pp. 737-744, Feb. 2015.

[11] L. O. Chua and S. M. Kang, "Section-wise piecewise-linear functions: Canonical representation, properties, and applications," *Proc. IEEE*, vol. 65, no. 6, pp. 915–929, Jun. 1977.

[12] L. O. Chua and A. C. Deng, "Canonical piecewise-linear representation," *IEEE Trans. Circuits Syst.**, vol. 35, no. 1, pp. 101-111, Jan. 1988.

[13] D. Muirhead, M. A. Imran, and K. Arshad, "Insights and approaches for low-complexity 5G small-cell base-station design for indoor dense networks," *IEEE Access*, vol. 3, pp. 1562-1572, Aug. 2015.

[14] D. Muirhead, M. A. Imran, and K. Arshad, "A survey of the challenges, opportunities and use of multiple antennas in current and future 5G small cell base stations," *IEEE Access*, vol. 4, pp. 2952-2964, May. 2016.

[15] R. N. Braithwaite, "A comparison of indirect learning and closed loop estimators used in digital predistortion of power amplifiers," in *IEEE MTT-S International Microwave Symposium*, pp. 1-4, May. 2015.

[16] H. Paaso and A. Mammela, "Comparison of direct learning and indirect learning predistortion architectures," in *IEEE International Symposium on Wireless Communication Systems*, pp. 309-313, Oct. 2008.

[17] L. Guan and A. Zhu, "Optimized low-complexity implementation of least squares-based model extraction for digital predistortion of RF power amplifiers," *IEEE Trans. Microw. Theory Techn.*, vol. 60, pp. 594–603, Mar. 2012.

[18] J. E. Cousseau, J. L. Figueroa, S. Werner, and T. I. Laakso, "Efficient nonlinear Wiener model identification using a complex-valued simplicial canonical piecewise linear filter," *IEEE Trans. Signal process.*, vol. 55, no. 5, pp. 1780-1792, May. 2007.

[19] R. G. Batruni, "Low-complexity nonlinear filters," U. S. 7613759B2, Nov. 3, 2009.

[20] L. Guan and A. Zhu, "Low-cost FPGA implementation of Volterra series-based digital predistorter for RF power amplifiers," *IEEE Trans. Microw. Theory Techn.*, vol. 58, no. 4, pp. 866-872, Apr. 2010.
[21] P. L. Gilabert, A. Cesari, G. Montoro, E. Bertran, and J.-M. Dilhac, "Multi-lookup table FPGA implementation of an adaptive digital predistorter for linearizing RF power amplifiers with memory effects," *IEEE Trans. Microw. Theory Techn.*, vol. 56, no. 2, pp. 372–384, Feb. 2008.
[22] J. E. Volder, "The CORDIC trigonometric computing technique," *IRE Trans. Electronic Computers*, vol. EC-8, no. 3, pp. 330-334, Sept. 1959.
[23] Y. H. Hu, "CORDIC-based VLSI architectures for digital signal processing," *IEEE Signal Process. Mag.*, vol. 9, no. 3, pp. 17–34, Jul. 1992.

Wenhui Cao (S'15) received the B.E. degree in automation from Beijing University of Chemical Technology, Beijing, China, in 2013 and is currently working toward the Ph.D. degree in University College Dublin (UCD), Dublin, Ireland.

She is currently with RF and Microwave Research Group, UCD. Her research interests include nonlinear behavioral modeling and linearization of RF power amplifiers, digital post-correction of high speed ADCs, and digital suppression of TX-induced interference in frequency-division duplexing (FDD) transceiver. She also has interests in high performance field-programmable gate-array (FPGA) implementation methodologies.

Anding Zhu (S'00-M'04-SM'12) received the B.E. degree in telecommunication engineering from North China Electric Power University, Baoding, China, in 1997, the M.E. degree in computer applications from Beijing University of Posts and Telecommunications, Beijing, China, in 2000, and the Ph.D. degree in electronic engineering from University College Dublin (UCD), Dublin, Ireland, in 2004.

He is currently an Associate Professor with the School of Electrical and Electronic Engineering, UCD. His research interests include high-frequency nonlinear system modeling and device characterization techniques with a particular emphasis on behavioral modeling and linearization of RF power amplifiers for wireless communications. He also has interests in high efficiency power amplifier design, wireless transmitter architectures, digital signal processing and nonlinear system identification algorithms.