Received 5 October 2022, accepted 19 October 2022, date of publication 28 October 2022, date of current version 7 November 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3218109

RESEARCH ARTICLE

A Sparse Neural Network-Based Power Adaptive DPD Design and Its Hardware Implementation

MASAAKI TANIO, NAOTO ISHII, AND NORIFUMI KAMIYA, (Member, IEEE)
NEC Corporation, Kawasaki, Kanagawa 211-8666, Japan
Corresponding author: Masaaki Tanio (m-tanio@nec.com)

ABSTRACT In this paper, an efficient neural-network-based adaptive DPD design which performs well under power varying conditions is presented. The DPD design is derived on the basis of the envelop time-delay neural network (ETDNN). The redefined ETDNN-DPD requires the part of parameter updates, which enables to adapt it to the rapid change of power amplifier (PA) distortion. Additionally, the redefined ETDNN-DPD also maintains the stability of the compensation performances under varying power condition while its structure is pruned by the structured pruning. Furthermore, to verify its practical use, we also propose the weight scaling technique, which reduces multiplications of the redefined ETDNN-DPD, and applied it to the implementation of the redefined ETDNN-DPD on FPGA. Compared FPGA-implemented ETDNN-DPD with FPGA-implemented conventional memory polynomial DPD, we verified that our proposed DPD achieved 3.2 dB better error vector magnitude (EVM) while lower hardware resource utilization at the fixed power level. Moreover, our proposed DPD kept better performance under the varying power condition only by the partial update of its parameters than memory polynomial DPD.

INDEX TERMS Digital predistortion, memory polynomial, FPGA implementation, neural network, pruning technique, physical model.

I. INTRODUCTION

Recent wireless systems, such as fifth-generation (5G) and beyond 5G wireless communication systems, are becoming complex and power hungry due to wider bandwidth and higher carrier frequency. Then, the nonlinear behaviors of power amplifiers (PAs) become the obstacle for the high-speed communication and energy efficient systems. To overcome the problem, digital predistortion (DPD) is one of the most effective techniques to compensate for the nonlinearity of PA.

Many DPD models based on Volterra series models including memory polynomial (MP) model [1] and the generalized MP (GMP) model [2], have been widely used. However, it is reported that recent efficient PA architectures, such as Doherty, envelope tracking and outphasing, are so complex that Volterra-based DPDs are not sufficient enough to compensate for their PA distortions due to their limited structures [3]. Recently, neural network (NN) has been considered as a promising method for DPD thanks to its model fitting capacity and many DPD models based on NN have been developed [4], [5], [6], [7], [8], [9]. Moreover, combining NN-based DPD with the pruning effectively reduces the computational complexity while keeping the performance [9], [10], [11], which brings NN-based DPDs close to installing in the wireless systems.

As for practical usage of DPDs, the distortion of a PA sometimes changes due to the power control, the thermal changes of a PA, etc. Thus, adaptive DPDs are often required to cope with the change of a PA distortion. Conventional approach for Volterra based DPD is that the DPD designer has selected and fixed the nonlinear order and the memory length, and only parameters are updated by using feedback signal from PA when the distortion of PA changes. The approach of NN-based DPDs is basically the same as that of Volterra-based DPDs, namely only parameters are updated for the adaptation [12], [13]. However, for NN-based DPDs, there are two big problems that prevent them from being practical

The associate editor coordinating the review of this manuscript and approving it for publication was Di Zhang.

usages: (i) The cost of parameter update (ii) The stability of the compensation performances under power varying conditions.

The cost of parameter update should be lower for the applications that require the frequent power control such as base station. In conventional Volterra based DPDs, their parameters can be updated by simple least-mean square method [1], which enables to rapid updates of the parameters, while NN based DPDs often suffer the high computational cost in the parameter updates due to the complex training of NNs. For instance, real valued time delay neural network (RVTDNN) uses the well-known Levenberg-Marquardt backpropagation algorithm for updating the weights and biases in NNs [4], [13] and the backpropagation requires the computation of the gradient at each layer, which increases the computational complexity dramatically. To reduce the cost of the parameter updates, the strategy to update the partial parameter have been proposed in CNN-based DPD [11]. However, only the parameters in the filter layer were fixed and those in the fully connected (FC) layer and the output layer should be updated, which still required the computation of the gradient.

The other issue for NN-based DPDs is the stability of the compensation performances under power varying conditions. To reduce the computational complexity while keeping its performance, pruning methods are often applied to the NN based DPDs [9], [10], [11]. However, changing the DPD structure by pruning sometimes leads to the degradation of compensation performance since the pruned DPD structure is calculated by using the training data under the limited condition. To validate the stability of pruned DPDs, pruning based on Bayesian framework for the GMP-DPD [14] demonstrated the robustness to changes in the power level. However, this pruning approach cannot apply to the NN-based DPDs that have multistage connections of neural networks and as far as we know, there is no report about the robustness of pruned NN-based DPDs to changes in the power level at the present time.

In order to overcome these problems for NN-based DPDs, in this paper, we present an efficient neural-network-based adaptive DPD design which performs well under power varying conditions. The DPD design is derived on the basis of the envelop time-delay neural network (ETDNN) [9]. The redefined ETDNN-DPD just needs the update of the partial parameters while keeping its connections for the adaptation of power varying conditions. This redefinition allows ETDNN-DPD to realize both the cost reduction of parameter update and its stable compensation performance under power varying conditions. In addition, we also propose the simple architecture of a neuron to reduce the calculation of multiplication and implement it on FPGA. Compared the redefined ETDNN-DPD with the conventional memory polynomial DPD on the condition that both DPDs are implemented on FPGA, we verified that the redefined ETDNN-DPD performs better error vector magnitude (EVM) despite its lower hardware resource utilization of FPGA at the fixed power level.

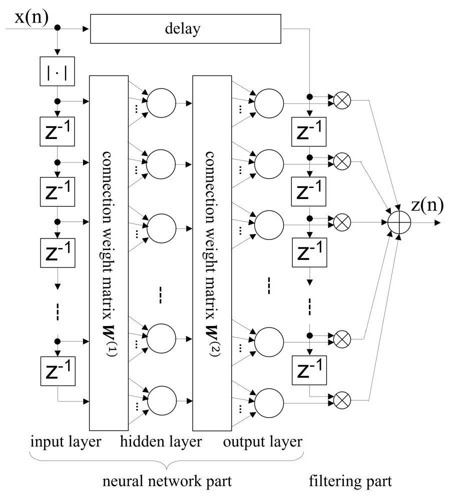

FIGURE 1. Block diagram of envelope time-delay neural network.

Furthermore, we also verified its stability of the compensation performances under the power varying condition only by the partial update of parameters.

The paper is organized as follows. In Section II, ETDNN model and its pruning method, which are proposed in [9], are introduced. In Section III, we implement deformed ETDNN-DPD on FPGA by using proposed weight scaling for the reduction of multiplication. In Section IV, experimental validations by FPGA implementation are presented in Section IV. Conclusions are given in Section V.

II. ENVELOPE TIME-DELAY NEURAL NETWORK FOR DIGITAL PRE-DISTORTION

A. ENVELOPE TIME-DELAY NEURAL NETWORK

We first give a brief review of the envelop time-delay neural network (ETDNN) presented in [9] for digital pre-distortion (DPD) design. Fig. 1 shows a block diagram of the ETDNN, which is composed of two parts: a neural network (NN) part and a filtering part. The NN part is a three-layer NN with $M+1$ real inputs, $M+1$ complex outputs, and N hidden layer neurons. The inputs are the magnitudes of $M+1$ consecutive samples $x(n), x(n-1), \ldots, x(n-M)$, which we denote in vector form by $|\boldsymbol{x}(n)|$. Let $\boldsymbol{y}^{(1)}(n) \in \mathbb{R}^N$, which is an N-dimensional and real-valued vector, and $\boldsymbol{y}^{(2)}(n) \in \mathbb{C}^{M+1}$, which is an $(M+1)$-dimensional and complex-valued vector, denote respectively the output signal vectors of the hidden and the output layers. Then they can be written in matrix form as follows:

$$\boldsymbol{y}^{(1)}(n) = \phi\Big(\boldsymbol{W}^{(1)}|\boldsymbol{x}(n)| + \boldsymbol{b}^{(1)}\Big) \tag{1}$$

$$\boldsymbol{y}^{(2)}(n) = \boldsymbol{W}^{(2)}\boldsymbol{y}^{(1)}(n) + \boldsymbol{b}^{(2)} \tag{2}$$

where $\boldsymbol{W}^{(1)} \in \mathbb{R}^{N\times(M+1)}$, $\boldsymbol{W}^{(2)} \in \mathbb{C}^{(M+1)\times N}$, $\boldsymbol{b}^{(1)} \in \mathbb{R}^N$ and $\boldsymbol{b}^{(2)} \in \mathbb{C}^{M+1}$ are the connection weight matrices and bias vectors of the hidden and output layers; and $\phi(\cdot)$ denotes the elementwise ReLU function. The output signal vector $\boldsymbol{y}^{(2)}(n)$ is used as the tap coefficients of the FIR filter in the

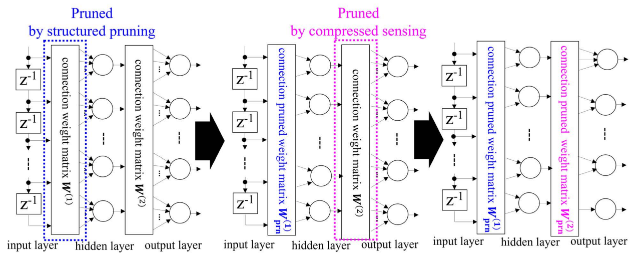
Pruned
by structured pruning
Pruned
by compressed sensing
z-1
connection weight matrix W(1)
connection weight matrix W(2)
connection pruned weight matrix W(1)prn
connection pruned weight matrix W(2)prn
input layer
hidden layer
output layer

FIGURE 2. Procedure of structured pruning for ETDNN.

filtering part of the ETDNN. Thus, the output signal $z(n)$ of the ETDNN of Fig.1 can be written as follows:

$$z(n) = \boldsymbol{y}^{(2)}(n)^T \boldsymbol{x}(n). \tag{3}$$

ETDNN described as (1) - (3) is also formulated by decomposing the connection matrices $\boldsymbol{W}^{(1)}$, $\boldsymbol{b}^{(1)}$, $\boldsymbol{W}^{(2)}$ and $\boldsymbol{b}^{(2)}$ as follows:

$$z(n) = \sum_{m=0}^{M} \left\{ \sum_{j=1}^{N} w_{j,m}^{(2)} \phi \left(\sum_{l=0}^{M} w_{l,j}^{(1)} |x(n-l)| + b_j^{(1)} \right) + b_m^{(2)} \right\} x(n-m) \tag{4}$$

where $w_{l,j}^{(1)}$ and $b_j^{(1)}$ are the coefficient weights and biases in $\boldsymbol{W}^{(1)}$ and $\boldsymbol{b}^{(1)}$, respectively and $w_{j,m}^{(2)}$ and $b_m^{(2)}$ are the coefficient weights and biases in $\boldsymbol{W}^{(2)}$ and $\boldsymbol{b}^{(2)}$, respectively. As is noted in [9], the ETDNN satisfies odd-parity and unitary phase constraints which should be complied within nonlinearity modeling for DPD [15], [16].

B. ETDNN WITH THE STRUCTURED PRUNING

In [9], we have presented a structured pruning technique [9] for reducing the computational complexity of the ETDNN. The structured pruning has two-step procedure as shown in Fig. 2.

In the first step, the connection matrix $\boldsymbol{W}^{(1)}$ is replaced to the pruned connection matrix $\boldsymbol{W}_{prn}^{(1)}$ that has the same size as $\boldsymbol{W}^{(1)}$ but most values are zero, which equals to be pruned, while keeping the lower value of the training error by optimizing the following equation:

$$\min_{\boldsymbol{W}_{prn}^{(1)}, \boldsymbol{b}^{(1)}, \boldsymbol{W}^{(2)}, \boldsymbol{b}^{(2)}} \sum_n |\tilde{z}(n) - z(n)|^2 \tag{5}$$

where $z(n)$ is the output signal of ETDNN, as shown in (3), that depends on the connection matrices $\boldsymbol{W}_{prn}^{(1)}$, $\boldsymbol{b}^{(1)}$, $\boldsymbol{W}^{(2)}$, $\boldsymbol{b}^{(2)}$ and $\tilde{z}(n)$ is the training signal obtained by iterative learning control (ILC) [17]. Then, the group-lasso learining [9] is applied to minimize (5). This algorithm satisfies the constraints that the number of nonzero row weight is equal to the predetermined fixed value of $|\sigma_j^{(1)}|$. It enables to extract the physically meaningful connection in $\boldsymbol{W}^{(1)}$.

In the next step, the connection matrices $\boldsymbol{W}^{(2)}$ and $\boldsymbol{b}^{(2)}$ are replaced to the pruned connection matrices $\boldsymbol{W}_{prn}^{(2)}$ and $\boldsymbol{b}_{prn}^{(2)}$, respectively. The pruned connection matrices $\boldsymbol{W}_{prn}^{(2)}$ and $\boldsymbol{b}_{prn}^{(2)}$ have the same size as $\boldsymbol{W}^{(1)}$ and $\boldsymbol{b}^{(1)}$, respectively, but most values are zero while keeping the lower value of the training error by optimizing the following equation:

$$\min_{\boldsymbol{W}_{prn}^{(2)}, \boldsymbol{b}_{prn}^{(2)}} \sum_n |\tilde{z}(n) - z(n)|^2 \tag{6}$$

where $z(n)$ is the output signal of ETDNN, as shown in (3), that depends on the connection matrices $\boldsymbol{W}_{prn}^{(1)}$, $\boldsymbol{b}^{(1)}$, $\boldsymbol{W}_{prn}^{(2)}$ and $\boldsymbol{b}_{prn}^{(2)}$. As is noted that $\boldsymbol{W}_{prn}^{(1)}$ and $\boldsymbol{b}^{(1)}$ in the previous step are used for the calculation of $z(n)$ in (6) and the compressed sensing algorithm [18] can be applied to solve (6). This procedure ensures the dramatical reduction of the computational complexity [9]. Moreover, in [9], we also clarified that 1 or 2 is enough for the number of nonzero row weight in $\boldsymbol{W}_{prn}^{(1)}$, namely $|\sigma_j^{(1)}| = 1$ or 2, from the viewpoint of minimization of the training error. For this reason, we focus on the ETDNN with the structured pruning whose $|\sigma_j^{(1)}|$ is 1 or 2 for FPGA implementation in the next section.

III. FPGA IMPLEMENTATION OF ENVELOPE TIME-DELAY NEURAL NETWORK WITH THE STRUCTURED PRUNING

In this section, ETDNN-DPD with the structured pruning under the restriction of the $|\sigma_j^{(1)}| = 1$ or 2 is implemented on FPGA. Generally speaking, the varying power condition requires the varying structure of ETDNN-DPD. It means that the structure of ETDNN should be changed by recalculating the structural pruning algorithm each time we change the power condition of PAs. It is not practical since hardware implemented DPDs are very hard to change their circuits. Even if the circuits can be changed corresponding to the power condition, we need to prepare many varieties of circuits and implemented them on the hardware in advance. It leads to increase the hardware resources.

To relax the requirements as above, we assume the following condition in ETDNN-DPD for practical usage.

Assumption 1: In case that the number of nodes in hidden layer (represented as N in (4) is large enough, when the output power of a PA is changed, the connection matrices $\boldsymbol{W}_{prn}^{(1)}$, $\boldsymbol{b}^{(1)}$ $\boldsymbol{W}_{prn}^{(2)}$ and $\boldsymbol{b}_{prn}^{(2)}$ in ETDNN-DPD must be updated while keeping the all zero values, namely pruned connections, in $\boldsymbol{W}_{prn}^{(1)}$, $\boldsymbol{b}^{(1)}$ $\boldsymbol{W}_{prn}^{(2)}$ and $\boldsymbol{b}_{prn}^{(2)}$ are kept to zero.

Assumption 2: In case that the number of nodes in hidden layer is large enough, when the output power of a PA is changed, the only connection matrices $\boldsymbol{W}_{prn}^{(2)}$ and $\boldsymbol{b}_{prn}^{(2)}$ in ETDNN-DPD should be updated and $\boldsymbol{W}_{prn}^{(1)}$ and $\boldsymbol{b}^{(1)}$ can be fixed.

Asumption 1 means that we do not need to change connections but need to update parameters on FPGA according to power change of PAs, which relaxes the hardwre requirement. Moreover, Asumption 2 also relaxes the computational complexity of parameter update since $\boldsymbol{W}_{prn}^{(2)}$ and $\boldsymbol{b}_{prn}^{(2)}$ are the coefficients at output layer and do not need to apply backpropagation for optimization. Note that the correctness of

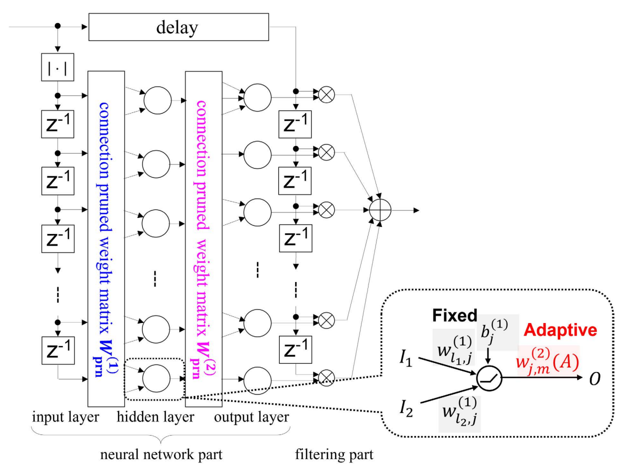

FIGURE 3. FPGA implementation of envelope time-delay neural network with the structured pruning.

these assumption is verified experimentally in the Section IV. Here, assuming that Assumptions 1 and 2 are correct, we consider the new architecture that enables further reduction of computational complexity.

Fig. 3 shows ETDNN-DPD with the structured pruning under $|\sigma_j^{(1)}| = 2$. Compared with Fig. 1 of the original ETDNN-DPD structure, the structured pruning reduces the connection of the neurons effectively. Moreover, Assumption 2 indicates the power varying operation only affects the weights $w_{j,m}^{(2)}(A)$ and biases $b_m^{(2)}(A)$ in (4), where A is the average amplitude of input signals. In other words, we can fix the weights $w_{l,j}^{(1)}$ and biases $b_j^{(1)}$ in (4). This restriction also contributes to reducing the hardware utilization.

Nevertheless, those drastic reduction realizes the practical implementation of ETDNN-DPD on FPGA, the hardware utilization of ETDNN-DPD is still large due to the multiplication by using weights. Then, to reduce multiplications on FPGA, a partial linear characteristics of ReLU, which is used as the activation function, can be utilized by combining with the fixed weights $w_{l,j}^{(1)}$ and biases $b_j^{(1)}$. We introduce the formula transformation of ReLU function, which leads to reduce multiplications on FPGA without losing accuracy in the next section. Moreover, to accelerate the operation of the ETDNN-DPD while keeping its compactness, we also propose the architecture of both the neuron element (NE) in the hidden layer and the linear element (LE) in the output layer.

A. WEIGHTS SCALING FOR REDUCTION OF MULTIPLICATION

The activation function ReLU is linear when input value is 0 and above. This partial linear characteristic leads the reduction of a multiplication for a neuron. At first, considering the number of input connection is two, we define ReLU input including weights multiplication and a bias addition in the following:

$$R_{in}^{(1)} := w_{l_1,j}^{(1)} I_1 + w_{l_2,j}^{(1)} I_2 + b_j^{(1)} \tag{7}$$

Then, the ReLU output O is obtained by multiplying the weight $w_o^{(2)}$ as follows:

$$O = \begin{cases} w_{j,m}^{(2)}(A) R_{in}^{(1)} & (R_{in}^{(1)} \geq 0) \\ 0 & (R_{in}^{(1)} < 0) \end{cases} \tag{8}$$

Assuming $w_{l_1,j}^{(1)} \neq 0$, (8) is formulated by multiplying $1/|w_{l_1,j}^{(1)}|$ as:

$$O = \begin{cases} |w_{l_1,j}^{(1)}| w_{j,m}^{(2)}(A) \dfrac{R_{in}^{(1)}}{|w_{l_1,j}^{(1)}|} & \left(\dfrac{R_{in}^{(1)}}{|w_{l_1,j}|} \geq 0\right) \\ 0 & \left(\dfrac{R_{in}^{(1)}}{|w_{l_1,j}|} < 0\right) \end{cases} \tag{9}$$

where

$$\frac{R_{in}^{(1)}}{|w_{l_1,j}^{(1)}|} = \begin{cases} I_1 + \dfrac{w_{l_2,j}^{(1)}}{|w_{l_1,j}^{(1)}|} I_2 + \dfrac{b_j^{(1)}}{|w_{l_1,j}^{(1)}|} & (w_{l_1,j}^{(1)} > 0) \\ -I_1 + \dfrac{w_{l_2,j}^{(1)}}{|w_{l_1,j}^{(1)}|} I_2 + \dfrac{b_j^{(1)}}{|w_{l_1,j}^{(1)}|} & (w_{l_1,j}^{(1)} < 0) \end{cases} \tag{10}$$

Compared (9)-(10) with (7)-(8), these formulation change the weights $w_{l_1,j}^{(1)}$, $w_{l_2,j}^{(1)}$, $w_{j,m}^{(2)}(A)$ and biases $b_j^{(1)}$ to the weights $w_{l_1,j}^{(1)}/|w_{l_1,j}^{(1)}|$, $w_{l_2,j}^{(1)}/|w_{l_1,j}^{(1)}|$, $|w_{l_1,j}^{(1)}|w_{j,m}^{(2)}(A)$ and biases $b_j^{(1)}/|w_{l_1,j}^{(1)}|$, respectively, while keeping the input of I_1 and I_2. Then, weights $w_{l_1,j}^{(1)}/|w_{l_1,j}^{(1)}|$, $w_{l_2,j}^{(1)}/|w_{l_1,j}^{(1)}|$ and biases $b_j^{(1)}/|w_{l_1,j}^{(1)}|$ are fixed since $w_{l_1,j}^{(1)}$, $w_{l_2,j}^{(1)}$ and $b_j^{(1)}$ are fixed by Assumption 2. In short, the scaling of $1/|w_{l_1,j}^{(1)}|$ is calculated by PC and the scaled values are fixed when they are implemented on FPGA. On the other hand, weights $|w_{l_1,j}^{(1)}|w_{j,m}^{(2)}(A)$ are directly calculated by using the measured error as described later. In other word, the multiplication of $|w_{l_1,j}^{(1)}|$ and $|w_{j,m}^{(2)}(A)$ are not needed on FPGA. Thus, these formula transformation just simplifies the multiplication of weights $w_{l_1,j}^{(1)}$ to 1 or -1 $(=w_{l_1,j}^{(1)}/|w_{l_1,j}^{(1)}|)$ on FPGA, which leads the furthermore reduction of hardware utilization. Note that we also apply the same formula transformation in case that the number of input connection is one, shown in the left side of Fig. 4.

B. ACCELERATION OF FPGA IMPLEMENTATION OF ETDNN-DPD WITH THE STRUCTURED PRUNING

The top-level overview of the redefined ETDNN-DPD with the structured pruning is shown in Fig. 5. It consists of delay bank1-2, neuron elements (NEs), linear elements (LEs) and complex FIR are corresponding to input layer, hidden layer, output layer and phase filter in Fig. 1, respectively. Adaptive parameters $w_{j,m}^{(2)}(A)$ and $b_m^{(2)}(A)$ are stored in RAM which can be updated from outside. The number of inputs in NE is one or two by applying the structured pruning [9] before the implementation on FPGA. The number of inputs in LE is defined by compressed sensing algorithm [18]. Delaybank1 corresponds

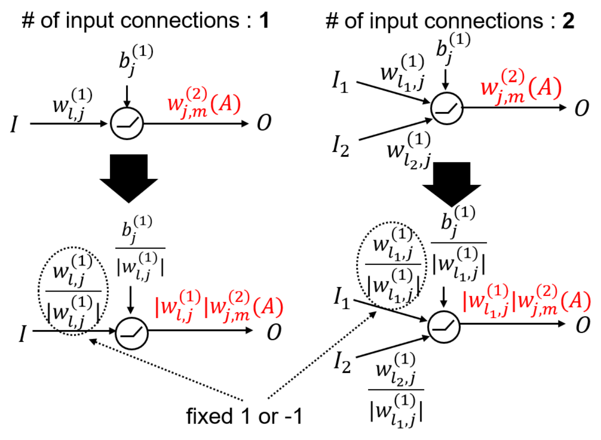

FIGURE 4. Weights scaling in a neuron for $|\sigma_j^{(1)}| = 1$ and 2.

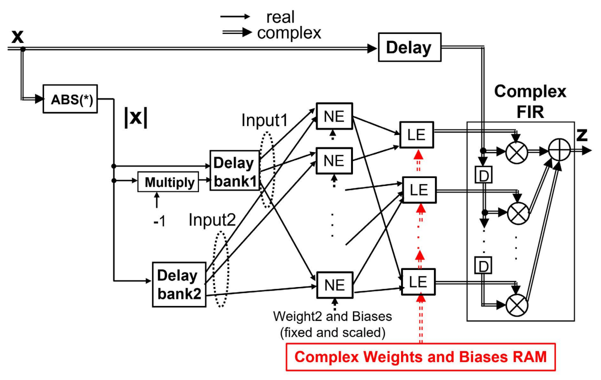

FIGURE 5. Top-level overview of the redefined ETDNN-DPD with the structured pruning.

to the delay taps for amplitude $|x(k)|$ to make the first input for neurons. As is described before, the weights of the first input are uniformly scaled to 1 or -1, which are realized by keeping or inverting a sign bit. On the other hand, the delay taps of the second input for neurons are integrated with delaybank2. The scaled weights of the second input $w_{l_2,j}^{(1)}/|w_{l_1,j}^{(1)}|$ are different individually and their multiplications are included in NE.

To realize high speed operation and low hardware utilization, hardware architectures of NE, LE and complex FIR are also essential. As is well known, complex FIR is commonly used and easily refer such as transposed direct form [19], whereas NE and LE are so specific that specific architectures are needed. In the next and one next subsection, we propose the optimized NE and LE structure in order.

1) HIGH SPEED AND COMPACT NEURON ELEMENT

Fig. 6 shows the block diagram of an NE. Fig. 6 to the left shows the case that the number of inputs for a neuron is one. Thanks to the weight scaling, we just use input1, which is pre-multiplied by 1 or -1, as input. Then, an input1 is added by a fixed and scaled bias and fed to a ReLU, which is easy to be accelerated. On the other hand, the case that the number

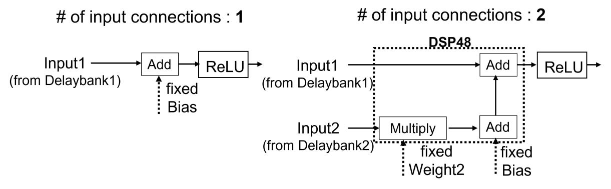

FIGURE 6. Block diagram of a neuron element with the structured pruning for $|\sigma_j^{(1)}|$=1 and 2.

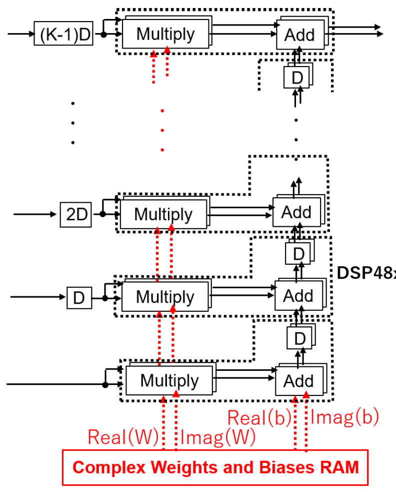

FIGURE 7. Block diagram of a linear element.

of inputs for a neuron is two, which is shown in Fig. 6 to the right, needs the additional multiplication and adder for input2. However, by utilizing Xilinx DSP48 unit [20], which has a multiplication and two adders, the calculation of input for a ReLU is executed by only one DSP48. In short, both cases are easy to be accelerated with minimum utilization.

2) HIGH SPEED AND COMPACT LINEAR ELEMENT

Fig. 7 shows the block diagram of a LE, where K is the number of inputs for a LE. The complex multiplication is separated to a real part and an imaginary part to reduce the multiplication. Furthermore, pipeline adders are adopted to both a real and an imaginary part for acceleration. The structured is realized by using cascaded Xilinx DSP48 [20], which enables high speed operation. Note that the difference of the latency between LEs, which is caused by the difference of the number K due to the non-uniform number of selected connections by compressed sensing algorithm, are cancelled by adding the minimum latency in output of LEs for synchronization as complex FIR input.

IV. MEASURED RESULTS

Fig. 8 shows the experimental setup including an FPGA and RF systems. The Xilinx ZCU111 evaluation kit was used for the calculation of DPD and output / input the signal for RF systems by using RF-DAC / RF-ADC, respectively. We used a 1024-QAM 46.08-MHz bandwidth single carrier

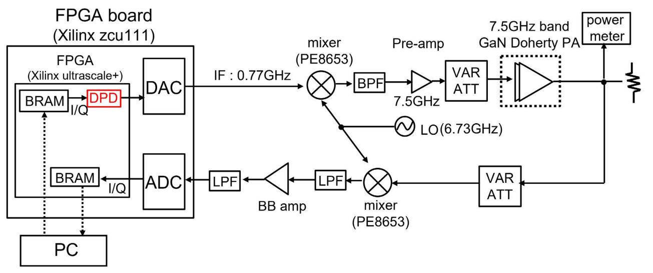

FIGURE 8. Measurement set up of 46-MHz Single Carrier in 7.5 GHz band.

signal with 368.64 MSa/s (8 times over-sample) and peak-to-average power ratio (PAPR) of 8.9 dB. The DPD output signal from RF-DAC was fed to the mixer (Pasternack, PE8653) and up-converted to 7.5 GHz and up-converted signal is fed to a bandpass filter, a pre-amp, and a variable attenuator in order and finally fed to a 7.5-GHz-band GaN PA, which consisted of a driver amplifier and Doherty PA with a 30 W peak output power. The PA output was fed to a down converter (Pasternack, PE8653) with a 6.73-GHz local frequency. The PA output was then down-converted from 7.5 GHz to 770 MHz, and this down converted signal was digitalized with an RF-ADC in ZCU111. After the digital down-conversion from intermediate frequency (IF) band of 770 MHz to baseband on the FPGA, finally, the I and Q signals were acquired by the BRAM and sent to the PC for the calculation of DPD parameter update.

To obtain supervised data for the DPD training, we applied iterative learning control (ILC) [17], which can perfectly compensate for the non-modeled distortion for the limited signal length (262,144 samples in this measurement) by updating the input sampling iteratively. The average PA output power was fixed at 4.4 W in each iteration of ILC. Test data, which is different from the training data but has the same sample numbers and the same PAPR of 8.9 dB, is used for the validation. Using the training data and the supervised data, the structures of pruned ETDNN-DPDs are calculated by PC and implemented on FPGA. Then, the FPGA-implemented DPD circuits are mounted in the DPD block on FPGA in Fig. 8 for real-time calculation.

The performance comparison of the other DPDs can be applied by replacing the DPD block in Fig. 8. In the next subsection, we will explain how to select the DPD, which should be compared with ETDNN-DPD, by using training data and supervised data.

A. SELECTION OF DPDs FOR FPGA-IMPLEMENTATION

To select DPDs to be implemented on FPGA, firstly, we use the measurement data of 7.5-GHz-band GaN PA, especially supervised data obtained by ILC [17]. Then, floating point operations (FLOPs) [21] is used as the indicators of computational complexity. Additionally, normalized mean square error (NMSE) [22] is also used as indicators of remaining distortion. In particular, the training NMSE in this measurement

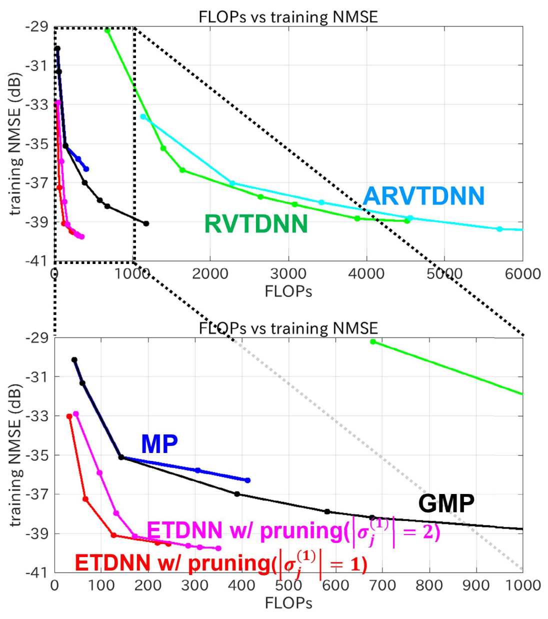

FIGURE 9. Relationship between FLOPs and training NMSE for MP, GMP, RVTDNN, ARVTDNN, and ETDNN w/ the structured pruning ($|\sigma_j^{(1)}| = 1$ and 2) for the supervised data of the 7.5-GHz-band GaN Doherty PA.

is referring to the error between supervised signal and DPD output signal.

ETDNN-DPD [9] with pruning ($|\sigma_j^{(1)}| = 1, 2$) was compared with conventional DPDs of MP [1], GMP [2], RVTDNN [4] and ARVTDNN [5]. For the fair comparison, the number of memory taps M for ETDNN-DPD, RVTDNN-DPD, and ARVTDNN-DPD was fixed to 10, the optimal value. N in ETDNN (4) was set as 200. In RVTDNN, we set two layers whose numbers of the neurons were varied from 10 to 50. and the best combinations of parameters were chosen from the viewpoint of the trade-off between FLOPs and training NMSE. In ARVTDNN, we set one layer whose number of the neurons was varied from 10 to 50 and $|x(k-m)|$, $|x(k-m)|^2$ and $|x(k-m)|^3$ were added as the amplitude terms to the input layer. For optimizations of the neural networks based DPDs, we used Adam [23]; the epoch number N_{ep} was 4000, and the batch size was 320. Structured pruning [9] efficiently restricts the number of input connections $|\sigma_j^{(1)}| = 1$ and 2 in ETDNN-DPD. The trade-off between FLOPs and training NMSE was controlled by varying the number of non-zero values in LS-OMP [18]. In this case, we found that the training NMSEs were saturated at 40 non-zero values for both $|\sigma_j^{(1)}| = 1$ and $|\sigma_j^{(1)}| = 2$ in LS-OMP. Finally, the coefficients of MP and GMP were optimized by using the least squares method. For a fair comparison, the parameters of the memory taps M, cross terms D, and the order of the polynomial P in MP and GMP were varied, and the best combinations of parameters were chosen from the viewpoint of the trade-off between FLOPs in the same manner as [21]. Note that we used the same FLOPs estimation as [21] for MP and GMP.

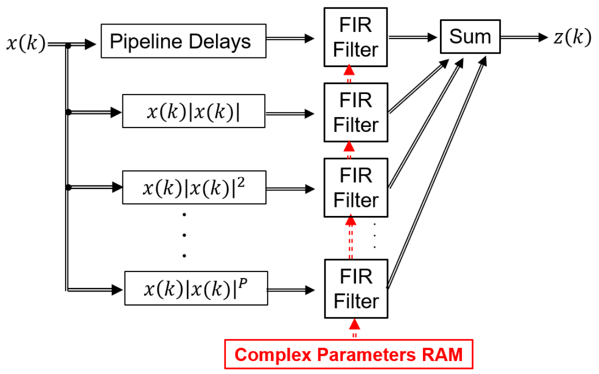

FIGURE 10. Hardware architecture of memory polynomial based on [12] including even order polynomials.

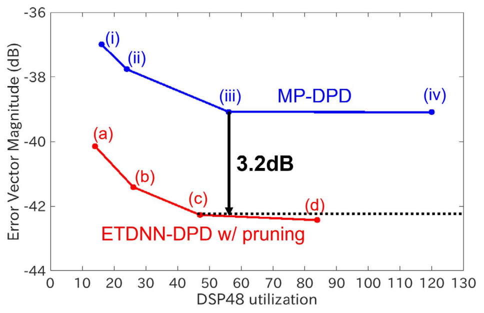

FIGURE 11. Relationship between the utilization of DSP48 and EVM for FPGA-implemented MP-DPD and ETDNN-DPD with pruning.

Fig. 9 shows the relationship between FLOPs and training NMSE. As we can see, in this measurement setting, the training NMSE of ETDNN-DPD with pruning ($|\sigma_j^{(1)}| = 1$) is almost the same as that of ETDNN-DPD with pruning ($|\sigma_j^{(1)}| = 2$), while FLOPs of ETDNN-DPD with pruning ($|\sigma_j^{(1)}| = 1$) is lower than that of ETDNN-DPD with pruning ($|\sigma_j^{(1)}| = 2$). Note that this performance is different from [9] since the distortion of 7.5-GHz GaN Doherty PA differs greatly from that of 3.5-GHz GaN Doherty PA used in [9], which can be seen by the AM-AM and AM-PM distortions of 7.5GHz GaN Doherty PA shown in Fig. 13. On the other hand, MP-DPD is the most efficient in all the conventional DPDs if we limit the FLOPs lower than 300 which is comparable to the maximum FLOPs of ETDNN-DPD with pruning ($|\sigma_j^{(1)}| = 1$).

From these results, in this paper, we focus on implementing MP-DPD and ETDNN-DPD with pruning ($|\sigma_j^{(1)}| = 1$) on FPGA. For the fair comparison, both DPDs are implemented with the similar fixed-point precision. In particular, to suppress the NMSE degradations of DPD output due to the limited fixed-point precision under 0.2 dB, 16 bit and 20 bit width were used for the neuron elements and the weights in ETDNN-DPD with pruning ($|\sigma_j^{(1)}| = 1$), respectively, and 16 bit and 21 bit width were used for the polynomial calculations and the coefficients in MP-DPD, respectively. Then, an efficient MP-DPD architecture is applied for FPGA implementation shown in Fig. 10. Note that Fig. 10 is based on MP-DPD in [12] and the difference between Fig. 10 and MP-DPD in [12] is that even order polynomials are included in Fig. 10 according to the commonly used MP model [1].

B. PARAMETER UPDATE OF FPGA-IMPLEMENTED DPDs

The parameter update of both MP-DPD and ETDNN-DPD are applied by the iterative learning control (ILC) scheme [17]. In the ILC scheme, the error signal $e(n)$ between input signal $x(k)$ and PA output signal $v(k)$ is calculated as follows:

$$e(n) = x(n) - \frac{1}{G}v(n) \tag{11}$$

where G is the gain of PA and by scaling of $1/G$, the average power of PA output signal becomes the same value as that of input signal. Then, using the error vector $\boldsymbol{e}(n)$ defined as Q consecutive samples $\boldsymbol{e}(n) := [e(n), e(n-1), \ldots, e(n-Q+1)]^T$, the parameters of $\boldsymbol{b}^{(2)}$ and $\boldsymbol{W}^{(2)}$ are updated as the following iterations.

$$[\boldsymbol{b}^{(2),i+1}\ \boldsymbol{W}^{(2),i+1}] = [\boldsymbol{b}^{(2),i}\ \boldsymbol{W}^{(2),i}] - \beta(\boldsymbol{X}^H\boldsymbol{X})^{-1}\boldsymbol{X}^H\boldsymbol{e}(n) \tag{12}$$

where i is the iteration number. β is a step size set lower than 1. $\boldsymbol{X}$ is the matrix of DPD model. In case of MP-DPD, $\boldsymbol{X}$ is defined from MP-DPD model [1] as follows:

$$\boldsymbol{X} = \big(\boldsymbol{x}(n)\ \ |\boldsymbol{x}(n)| \odot \boldsymbol{x}(n) \ldots |\boldsymbol{x}(n-M)|^P \odot \boldsymbol{x}(n-M)\big) \tag{13}$$

and in case of ETDNN-DPD with pruning, $\boldsymbol{X}$ is derived from (1) - (3) as

$$\boldsymbol{X} = \big(\boldsymbol{x}(n)\ \ \phi(\boldsymbol{W}^{(1)}|\boldsymbol{x}(n)| + \boldsymbol{b}^{(1)}) \odot (\boldsymbol{x}(n)\ \ \boldsymbol{x}(n) \ldots \boldsymbol{x}(n))\big) \tag{14}$$

Note that as we explained in the Section II, we only update the weights $w_{j,m}^{(2)}(A)$ and $b_m^{(2)}(A)$ and other parameters are fixed thanks to Assumption 1 and 2. It eases the parameter update without using the back propagation except for other NN-based DPDs. Moreover, additionally note that we can also use multi-objective optimization techniques [25], [26] for maximum RF output power by combining the optimization of the parameters in (12) with that of output power.

C. MEASUREMENT RESULTS UNDER THE FIXED POWER OPERATION

Fig. 11 show the relationship between the utilization of DSP48 and EVM for FPGA-implemented MP-DPD and ETDNN-DPD with pruning. To check the trade-off between hardware utilization and EVM, we implemented 4 types of DPD on FPGA for both MP-DPDs (i - iv) and ETDNN-DPDs with pruning (a - d) whose parameters are optimized by using the equations (12), (13) and (12), (14), respectively. we can see that ETDNN-DPD required the fewer DSP48s than MP-DPD did for the same EVM. This means that

TABLE 1. FPGA utilization of MP-DPD and ETDNN-DPD with pruning.

MP-DPD	(i)$P2M1$	(ii)$P3M1$	(iii)$P4M3$	(iv)$P6M5$
DSP48	16	24	56	120
LUP	569	705	1384	2745
LUTRAM	128	164	344	704
FF	1123	1432	2940	5913
Latency	30	31	32	34
ETDNN-DPD with pruning	(a)$cs5$	(b)$cs10$	(c)$cs20$	(d)$cs35$
DSP48	14	26	47	84
LUP	442	620	779	1051
LUTRAM	92	144	179	240
FF	851	1357	1929	2867
Latency	32	33	37	41

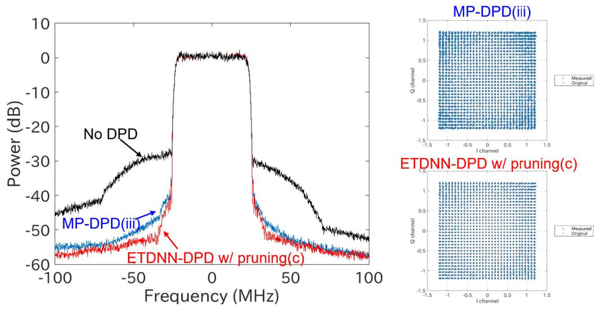

FIGURE 12. Output spectrum for MP-DPD and ETDNN-DPD with pruning.

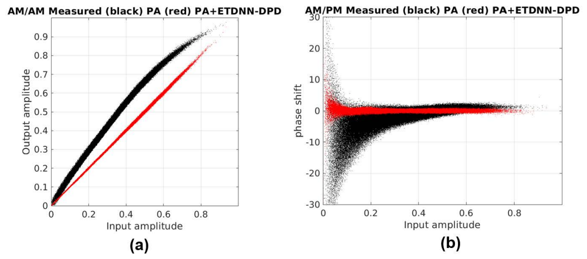

FIGURE 13. Characteristics of (a) AM-AM and (b) AM-PM without DPD (black) and with ETDNN-DPD with pruning (red).

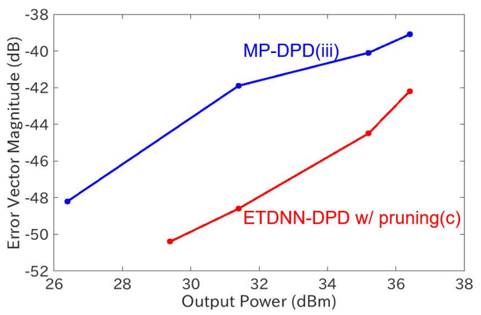

FIGURE 14. Relationship between output power and EVM with MP-DPD and ETDNN-DPD with pruning.

ETDNN-DPD enabled a compact DPD modeling. In particular, compared ETDNN-DPD (iii) with MP-DPD (c), which are near saturation points from the viewpoint of EVM, ETDNN-DPD (iii) performs 3.2-dB better EVM than MP-DPD (c).

Table 1 summarizes the detailed hardware utilization of FPGA-implemented DPDs. The utilization of DSP48 is the same as in Fig. 11. As is shown in Table 1, ETDNN-DPD with pruning (a - d) required the fewer not only DSP48 but also other IPs (LUT, LUTRAM and FF) than MP-DPD (i - iv), while ETDNN-DPDs (a - d) performed better EVM than MP-DPDs (i - iv), respectively.

The output spectrums and constellation corresponding to (c) and (iii) are also shown in Fig. 12. The spectrum leakage of ETDNN-DPD with pruning was lower than that of MP-DPD, while the required hardware utilization of ETDNN-DPD with pruning was lower than that of MP-DPD. Moreover, constellation also showed that in-band distortion of ETDNN-DPD with pruning was also lower that of MP-DPD.

D. MEASUREMENT RESULTS UNDER VARYING POWER OPERATION

Fig. 14 shows the relationship between output power and EVM under power-level changes. We choose MP-DPD (iii) and ETDNN-DPD with pruning (c) which have similar hardware utilization for comparison. Then, output powers of both DPDs are swept until EVM reaches about -50dB. The parameters of MP-DPD (iii) and ETDNN-DPD with pruning (c) are updated by using the equations (12), (13) and (12), (14) at each power level, respectively. Note that the computational complexity of (13) and (14) are almost the same and 5 iterations are applied for both DPDs. Compared ETDNN-DPD with pruning (c) with MP-DPD (iii), ETDNN-DPD with pruning (c) keeps better EVM for all power range. It verifies the ETDNN-DPD with pruning is robust to changes in the power level only by the partial update of its parameters. In other words, the correctness of Assumption 1 and 2 is proved by the experimental results. Note that we also discuss these results to give the theoretical interpretation in Appendix. Finally, these results demonstrate that the redefined ETDNN-DPD is a promising candidate for 5G and beyond 5G wireless transmitters.

V. CONCLUSION

In this paper, we have redefined the hardware-efficient ETDNN-DPD which requires the low hardware resource and additionally maintains the stability of the compensation performances under varying power condition. We have also proposed the weight scaling technique to reduce the multiplication for the FPGA implementation. Compared FPGA-implemented proposed DPD with FPGA-implemented conventional memory polynomial DPD, we verified that our proposed DPD achieved 3.2-dB better EVM while lower hardware resource utilization at the fixed power level. Furthermore, our proposed DPD kept better performance under the varying power condition only by the partial update of parameters.

APPENDIX - THEORETICAL INTERPRETATION OF ETDNN-DPD UNDER THE POWER VARYING CONDITION

In the Appendix, we give the physical interpretation to the ETNDNN-DPD under varying power condition. At first, we consider the GMP model that has theoretically interpretation under the power varying condition of PA by including the effects of varying power as follows [24]:

$$z_{GMP}(A;n) = \sum_{p=0}^{P}\sum_{m=0}^{M}\sum_{l=-L_m^{low}}^{L_m^{up}} \hat{h}_{m,l,p}(A)|x(n-l)|^p \times x(n-m) \quad (15)$$

where A is the average amplitude of input signals. When we set $L_m^{low} = 0$ and $L_m^{up} = M$, the GMP model (15) can be rewritten by using the notation in Section II as follows:

$$z_{GMP}(A;n) = \boldsymbol{g}(A;n)^T \boldsymbol{x}(n) \quad (16)$$

where $\boldsymbol{g}(A;n) \in \mathbb{C}^N$ can be represented as

$$\boldsymbol{g}(A;n) = \boldsymbol{G}(A)\begin{pmatrix} \boldsymbol{1} \\ |\boldsymbol{x}(n)| \\ |\boldsymbol{x}(n)|^2 \\ \vdots \\ |\boldsymbol{x}(n)|^P \end{pmatrix} \quad (17)$$

for some $\boldsymbol{G}(A) \in \mathbb{C}^{(M+1)\times(M+1)(P+1)}$. By the universal approximation theorem, for $p = 0, 1, 2, \ldots, P$, $|x|^p$ can be approximated as

$$|x|^p \sim \boldsymbol{w}_p^{(2)T}\phi(\boldsymbol{w}_p^{(1)}|x| + \boldsymbol{b}_p^{(1)}) + b_p^{(2)} \quad (18)$$

where $\boldsymbol{w}_p^{(1)}, \boldsymbol{w}_p^{(2)}, \boldsymbol{b}_p^{(1)} \in \mathbb{R}^{N_\infty}$ and $b_p^{(2)} \in \mathbb{R}$, and N_∞ is a sufficiently large positive integer. Thus, the following approximation holds for some $\boldsymbol{W}_G^{(1)} \in \mathbb{R}^{N_\infty\times(M+1)}$, $\boldsymbol{W}_G^{(2)} \in \mathbb{R}^{(M+1)(P+1)\times N_\infty}$, $\boldsymbol{b}_G^{(1)} \in \mathbb{R}^{N_\infty}$, and $\boldsymbol{b}_G^{(2)} \in \mathbb{R}^{(M+1)(P+1)}$:

$$\begin{pmatrix} \boldsymbol{1} \\ |\boldsymbol{x}(n)| \\ |\boldsymbol{x}(n)|^2 \\ \vdots \\ |\boldsymbol{x}(n)|^P \end{pmatrix} \sim \boldsymbol{W}_G^{(2)}\phi(\boldsymbol{W}_G^{(1)}|\boldsymbol{x}(n)| + \boldsymbol{b}_G^{(1)}) + \boldsymbol{b}_G^{(2)}. \quad (19)$$

We should note here that by (18) we may assume that the maximum row weight of $\boldsymbol{W}_G^{(1)}$ is one. Then, $\boldsymbol{W}_G^{(2)}$ and $\boldsymbol{b}_G^{(2)}$ are rewritten as

$$\begin{aligned} \boldsymbol{W}_G^{(2)}(A) &:= \boldsymbol{G}(A)\boldsymbol{W}_G^{(2)} \\ \boldsymbol{b}_G^{(2)}(A) &:= \boldsymbol{G}(A)\boldsymbol{b}_G^{(2)} \end{aligned} \quad (20)$$

where $\boldsymbol{W}_G^{(2)}(A) \in \mathbb{C}^{(M+1)\times N_\infty}$ and $\boldsymbol{b}_G^{(2)}(A) \in \mathbb{C}^{(M+1)}$ are depended on the average amplitude A. From Eq. (16) - (20), the output of GMP model derived from the view point of the physical aspect is summarized as

$$z_{GMP}(A;n) \sim \boldsymbol{W}_G^{(2)}(A)\phi(\boldsymbol{W}_G^{(1)}|\boldsymbol{x}(n)| + \boldsymbol{b}_G^{(1)}) + \boldsymbol{b}_G^{(2)}(A). \quad (21)$$

This equation implies two characteristics. One is that the GMP model can be seen as a special case of ETDNN with $|\sigma_j^{(1)}| = 1$. The other is that ETDNN with $|\sigma_j^{(1)}| = 1$ can compensate the distortions of the PAs under varying power condition by updating only $\boldsymbol{W}_G^{(2)}(A)$ and $\boldsymbol{b}_G^{(2)}(A)$, which supports the correctness of Assumption 1 and 2.

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] 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.

[3] A. Zhu, "Behavioral modeling for digital predistortion of RF power amplifiers: From Volterra series to CPWL functions," in *Proc. IEEE Topical Conf. Power Model. Wireless Radio Appl. (PAWR)*, Austin, TX, USA, Jan. 2016, pp. 1–4.

[4] M. Rawat and F. M. Ghannouchi, "A mutual distortion and impairment compensator for wideband direct-conversion transmitters using neural networks," *IEEE Trans. Broadcast.*, vol. 58, no. 2, pp. 168–177, Jun. 2012.

[5] D. Wang, M. Aziz, M. Helaoui, and F. M. Ghannouchi, "Augmented real-valued time-delay neural network for compensation of distortions and impairments in wireless transmitters," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 30, no. 1, pp. 242–254, Jan. 2019.

[6] Y. Zhang, Y. Li, F. Liu, and A. Zhu, "Vector decomposition based time-delay neural network behavioral model for digital predistortion of RF power amplifiers," *IEEE Access*, vol. 7, pp. 91559–91568, 2019.

[7] X. Hu, Z. Liu, X. Yu, Y. Zhao, W. Chen, B. Hu, X. Du, X. Li, M. Helaoui, W. Wang, and F. M. Ghannouchi, "Convolutional neural network for behavioral modeling and predistortion of wideband power amplifiers," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 33, no. 8, pp. 3923–3937, Aug. 2022.

[8] J. Sun, W. Shi, Z. Yang, J. Yang, and G. Gui, "Behavioral modeling and linearization of wideband RF power amplifiers using BiLSTM networks for 5G wireless systems," *IEEE Trans. Veh. Technol.*, vol. 68, no. 11, pp. 10348–10356, Nov. 2019, doi: 10.1109/TVT.2019.2925562.

[9] M. Tanio, N. Ishii, and N. Kamiya, "Efficient digital predistortion using sparse neural network," *IEEE Access*, vol. 8, pp. 117841–117852, 2020.

[10] Z. He and F. Tong, "Residual RNN models with pruning for digital predistortion of RF power amplifiers," *IEEE Trans. Veh. Technol.*, vol. 71, no. 9, pp. 9735–9750, Sep. 2022.

[11] Z. Liu, X. Hu, L. Xu, W. Wang, and F. M. Ghannouchi, "Low computational complexity digital predistortion based on convolutional neural network for wideband power amplifiers," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 69, no. 3, pp. 1702–1706, Mar. 2022.

[12] C. Tarver, A. Balatsoukas-Stimming, and J. R. Cavallaro, "Design and implementation of a neural network based predistorter for enhanced mobile broadband," in *Proc. IEEE Int. Workshop Signal Process. Syst. (SiPS)*, Oct. 2019, pp. 296–301.

[13] S. Yesil, C. Sen, and A. O. Yilmaz, "Experimental analysis and FPGA implementation of the real valued time delay neural network based digital predistortion," in *Proc. 26th IEEE Int. Conf. Electron., Circuits Syst. (ICECS)*, Nov. 2019, pp. 614–617.

[14] C. Crespo-Cadenas, M. J. Madero-Ayora, J. A. Becerra, and S. Cruces, "A sparse-Bayesian approach for the design of robust digital predistorters under power-varying operation," *IEEE Trans. Microw. Theory Techn.*, vol. 70, no. 9, pp. 4218–4230, Sep. 2022.

[15] E. G. Lima, T. R. Cunha, H. M. Teixeira, M. Pirola, and J. C. Pedro, "Base-band derived Volterra series for power amplifier modeling," in *IEEE MTT-S Int. Microw. Symp. Dig.*, Boston, MA, USA, Jun. 2009, pp. 1361–1364.

[16] E. G. Lima, T. R. Cunha, and J. C. Pedro, "A physically meaningful neural network behavioral model for wireless transmitters exhibiting PM–AM/PM–PM distortions," *IEEE Trans. Microw. Theory Techn.*, vol. 59, no. 12, pp. 3512–3521, Dec. 2011.

[17] J. Chani-Cahuana, P. N. Landin, C. Fager, and T. Eriksson, "Iterative learning control for RF power amplifier linearization," *IEEE Trans. Microw. Theory Techn.*, vol. 64, no. 9, pp. 2778–2789, Sep. 2016.

[18] M. Elad, *Pursuit Algorithms—Practice, in Sparse and Redundant Representations*. New York, NY, USA: Springer, 2010.

[19] A. Antoniou, *Digital Signal Processing: Signal, Systems, and Filters*. New York, NY, USA: McGraw-Hill, 2005.

[20] Xilinx. (2021). *DSP: Designing for Optimal Results High-Performance DSP Using Virtex-4 FPGAs*. [Online]. Available: https://www.xilinx.com/publications/archives/books/dsp.pdf

[21] A. S. Tehrani, H. Cao, T. Eriksson, M. Isaksson, and C. Fager, "A comparative analysis of the complexity/accuracy tradeoff in power amplifier behavioral models," *IEEE Trans. Microw. Theory Techn.*, vol. 58, no. 6, pp. 1510–1520, Jun. 2010.

[22] F. Mkadem, "Behavioural modeling and linearization of RF power amplifier using artificial neural networks," M.S. thesis, Dept. Elect. Eng., Univ. Waterloo, Waterloo, ON, Canada, 2010.

[23] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proc. ICLR*, 2015, pp. 1–10.

[24] C. Crespo-Cadenas, M. J. Madero-Ayora, and J. A. Becerra, "On the power level dependence of PA and DPD Volterra models," in *Proc. IEEE Topical Conf. RF/Microwave Power Model. Radio Wireless Appl. (PAWR)*, San Antonio, TX, USA, Jan. 2020, pp. 18–21.

[25] M. Mengozzi, G. P. Gibiino, A. M. Angelotti, C. Florian, and A. Santarelli, "GaN power amplifier digital predistortion by multi-objective optimization for maximum RF output power," *Electronics*, vol. 10, no. 3, p. 244, Jan. 2021.

[26] M. Mengozzi, A. M. Angelotti, G. P. Gibiino, C. Florian, and A. Santarelli, "Joint dual-input digital predistortion of supply-modulated RF PA by surrogate-based multi-objective optimization," *IEEE Trans. Microw. Theory Techn.*, vol. 70, no. 1, pp. 35–49, Jan. 2022.

MASAAKI TANIO received the B.E. and M.E. degrees in mathematical engineering from the University of Tokyo, Tokyo, Japan, in 2007 and 2009, respectively.

He joined at NEC Corporation, in 2009, where he is researching and developing RF power amplifiers and digital signal-processing techniques for wireless communications. He is currently a Principal Researcher with the Advanced Network Research Laboratories, NEC Corporation. His current research interests include neural-network-based digital pre-distortion techniques and delta-sigma modulations. He is a member of the Institute of Electronics, Information, and Communication Engineers (IEICE), Japan.

NAOTO ISHII received the B.E., M.E., and Ph.D. degrees in electrical engineering from Yokohama National University, in 1992, 1994, and 1997, respectively. He joined at NEC Corporation, Japan, in 1997, where he is currently a Lead Research Engineer with the Advanced Network Research Laboratories. His current research interests include signal processing and radio resource management for mobile wireless communication systems.

NORIFUMI KAMIYA (Member, IEEE) received the B.E. degree from Yokohama National University, in 1990, and the Ph.D. degree from the University of Tokyo, in 2000, all in electrical engineering.

He is currently a Senior Principal Researcher with the Advanced Network Research Laboratories, NEC Corporation, Japan. His current research interests include signal processing and coding for wireless and optical communication systems.