DPD基于LS和Transformer的应用

## 摘要

在通信领域中，传输信号中由于功放（PA）及其他器件的非线性特性和器件交调特性，导致天线端输出信号与功放输入的原始信号失真现象产生，以往的常用方法是采用数字预失真通过采用LS方法对功放模型输入输出的数据进行投影分析出其中的系数变换，然后进行逆向变换，通过滤波的方式来预处理输入数据以解决该问题。本文提出使用transformer的方式来更新DPD系数的处理过程，通过引入transformer来加快系数更新速率及提高误差的准确率。

## 介绍

在通信领域中，针对功放等模拟设备的非线性特性的特点传统的防止失真的方法是先发送训练序列，然后采集模拟端的输入数据及输出给天线端数据，并传回到数字端，让数字端通过LS算法推算出失真特性，提前在数字端进行逆向失真变换，【4】中提及到的方法是通过使用volterra级数及其导数来包含系统序列的多个非线性级数和过去的记忆级别，在FPGA中通过将不同记忆项多项式的计算结果都存储在查找表(LUT)中,从而高效地实现数字预失真处理（DPD），有利于提高峰均比（PAPR）。

在【1】中提及了一种基于自注意力机制的框架方法，用于描绘出输入输出的各个成分之间的全局关联性，再通过embedding编码及位置编码模块来生成出输入的相应位置的向量，然后通过与query、key、value空间之间进行线性空间变化，并且通过softmax来量化每行数据相关性的所占权重比例，从而获得最终相关性得分，从而推导出该元素与模型预训练的向量空间各个向量之间的关联性，最终获取最优相关向量，最通过解码模块，输出结果。

在【3】中提及了作者通过9个线性代数问题来探索transformer 的数值计算能力，及提及了一些不同的模型框架，并且使用训练的小型transformer模型展现了其可以以高达90%的精度来获取近似的数值解。在【2】中展现了传统训练的transformer文本中获取隐藏的线性向量维度信息，从而达到与最佳最小二乘方法能相抗衡的性能。

文献【4】中所提到的方法只处理了信号的有限的非线性特性和记忆特性，而通过引入transformer中的自注意力机制，从而实时获取PA的输入输出端的信号向量，从而在模型中与参考向量空间进行相关性对比，然后通过空间变换映射到参数空间的变换，输出更新后的参数。

本文当中首先展示了对文献【2】中的模型过程的构建和不经过DPD和经过DPD的PA后的信号性能特性，然后提出一种基于transformer实现的可能性。

## 模型

传统的DPD模型，通过采样到PA的输入端及输出端，来进行参数估计，如下图所示：

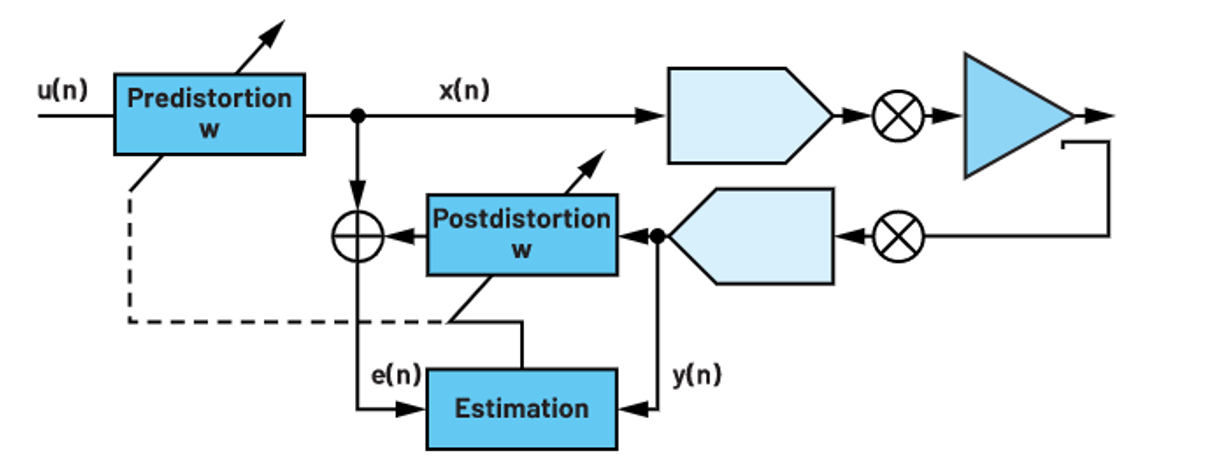


图1 . 传统的直接型DPD模型

误差处理：

预失真处理：

误差来估计参数

通常PA的理想模型为

即输出与输入是线性关系

现实使用中的PA是非线性模型

,

如下图所示：

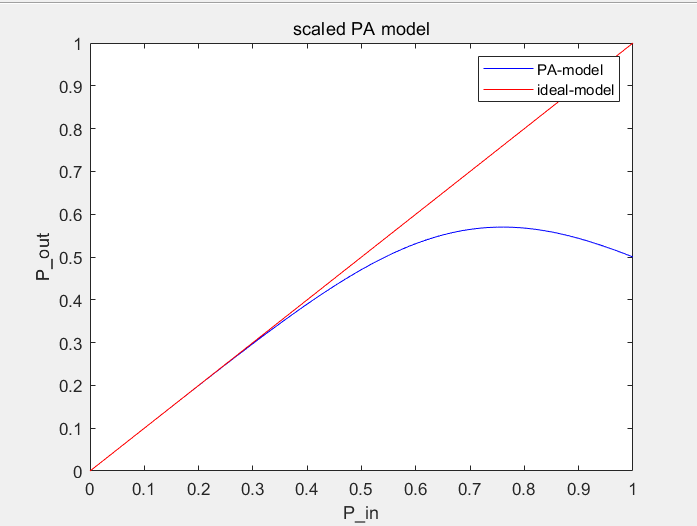
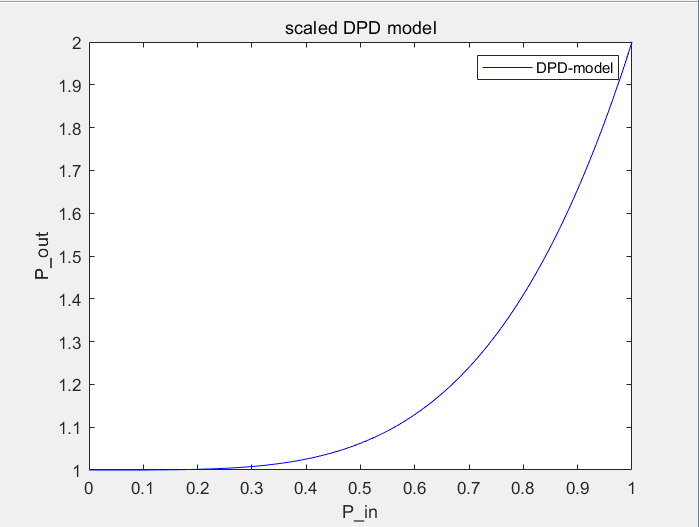
 

图1-1 PA的非线性放大特性 图1-2 数字预失真特性

针对于PA的非线性模型需要采用预失真的模型来恢复成线性化：

需要对PA的放大特性进行估计，对于非记忆非线性PA特性模型为

通常PA的输入信号为射频信号的实部：

那么非记忆PA的输出为

一般关注于载波附近的信号，时, 即p为奇数时的信号状况,

提取其等效基带的PA模型为

若考虑到记忆性效应及其中心载波的高次谐波频段的话

对于窗长度为N且记忆项为Q非线性度为 P的信号序列进行矩阵化：

由此求解出来的PA简化后的模型系数为：

使用LMS来进行DPD的参数估计,假设DPD的模型估计为

误差平方和为：

## 基本理论

对于LS的基本模型：

其中的矩阵求逆相当于除上了各个基向量的模的平方，以最小误差来投影Y到X的各个基向量上，并以各自不同系数（参考于基向量的模）的各个基向量线性组合而存在。

Transformer的基本模型：

注意力矩阵输出：, 分别作为Query、Key、Value及维度。

Softmax操作过程为： ,对矩阵的操作就是求出每行各元素的所占权重比例

## 方法

基于transformer进行DPD过程，关注于PA的输入前及输出的信号组成的向量，将该向量通过embedding层映射到模型内的向量，在embedding层将PA的输入经过DPD的模型以求出与线性放大的向量之间的距离向量，将该距离向量加上embedding生成的向量，然后采用自注意力机制求解出实际的向量空间与golden向量空间的相关性，通过输入

然后将与的不同的非线性级和记忆级别组合成一个向量组，如下：

*将与位置编码向量加在一起，组成,将传入到编码栈中进行多头注意力操作，然后进行层级归一化，再进行前馈层操作，将输入的向量进行线性空间变换出Q、K、V，从而从原有的向量空间获取到没看到的信息空间，然后将目标向量与Q、K、V向量空间之间的相关性，然后通过softmax来归一化为0~1的权重系数，来表明与向量之间的紧密程度。*

## 仿真测试

基于MATLAB的simulink上使用xilinx的工程块进行验证，用于仿真在FPGA上的算法实现效果，在FPGA上采用基于LUT表的方式来进行DPD操作，通过采集PA输入和输出的信号序列，进一步提取分析PA的放大的记忆特性及非线性特性。主要通过使用RAM进行存储各个幅度及各个不同非线性级数和记忆级数，再通过流水线加法来实现功能，如下图：

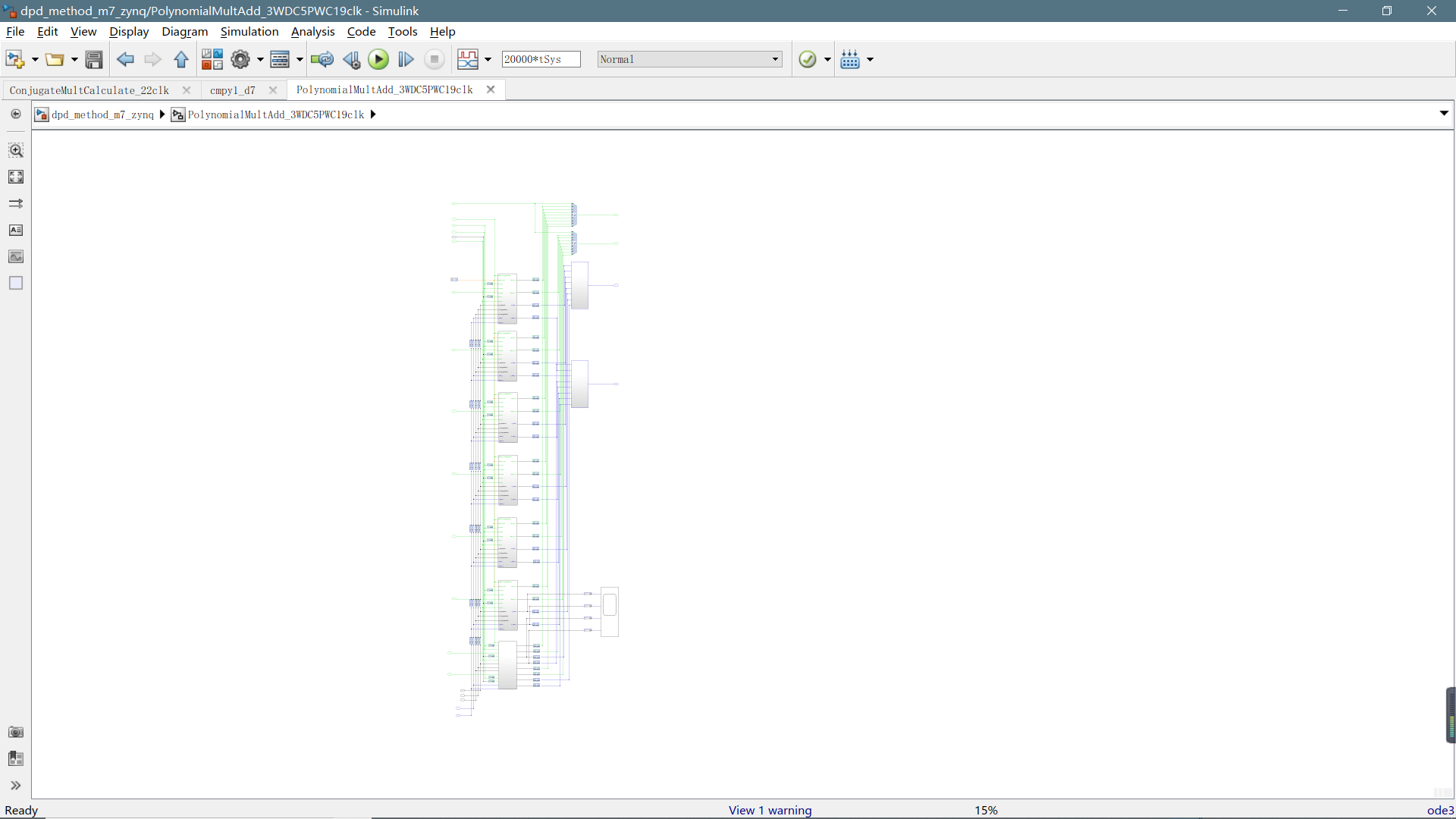


图1-3 DPD基于LUT实现过程仿真的simulink模型

由LS方法来线性化PA放大特性的DPD方法，其效果如下：

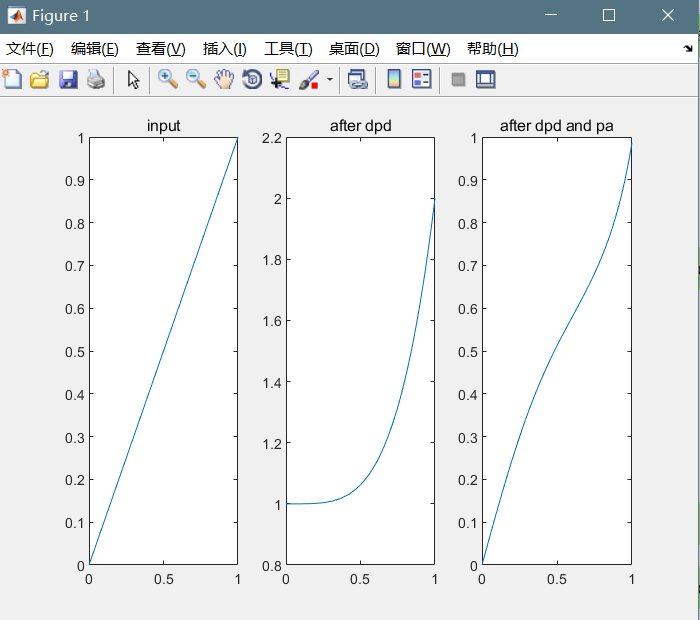


图1-4 原始数据及经过DPD和PA的增益图

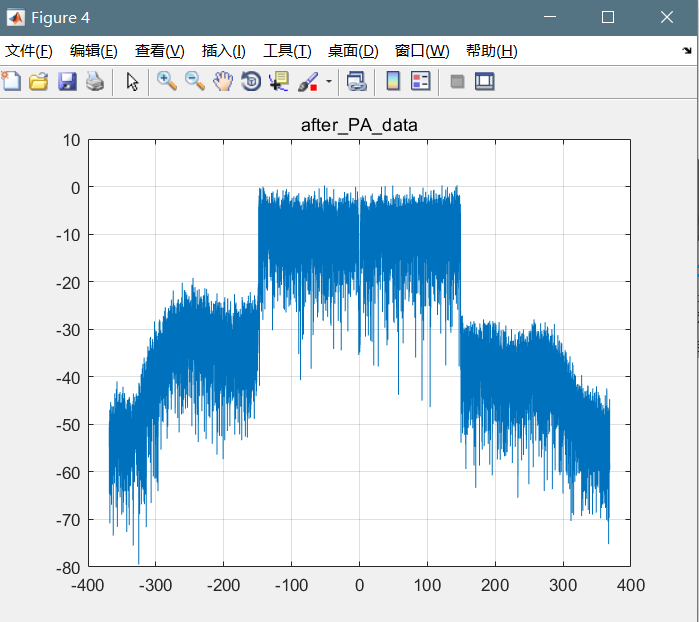
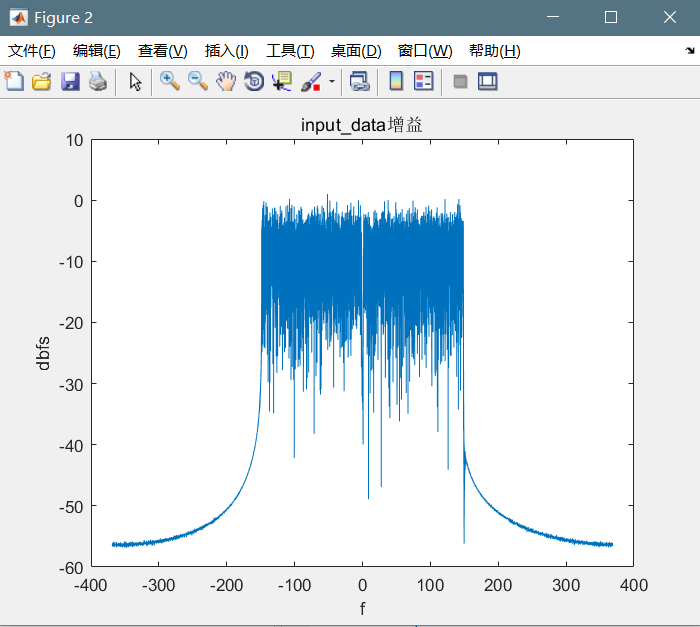


图1-5 输入的150M的带通信号及通过PA后的数据频谱

## 总结

在本文中主要分析了DPD的基于LS的方法实现，并作出相应的程序实验，最终可移植到FPGA上，并且提供了使用transformer应用于DPD场景的想法，还需要更深一步地探究embedding层的映射关系。后续还要深究对于transformer的模型搭建

## 参考文献

1. Attention Is All You Need
2. What Can Transformers Learn In-Context
3. Linear algebra with transformers
4. Digital prediction of power amplifiers using look-up table methods with memory effects for LTE wireless systems.

## 附录

代码 dpd\_lut\_matlab\_compa.m

global INPUT\_WIDTH LUT\_LUT LUT\_DUAL\_MODE LUT\_AN LUT\_BN Depth LUT\_N COMPARE

LUT\_N=84;

Depth = 12;

INPUT\_WIDTH = 16;

COMPARE = 1;

load('LUT.mat');

%% lut\_init

data\_lut\_96 = textread('./src\_0506/FPGA\_LUT.txt','%s'); %¸ßqµÍi

data\_lut\_96\_char=char(data\_lut\_96);

data\_lut\_96\_dec = hex2dec(data\_lut\_96\_char);

data\_lut\_q = floor(data\_lut\_96\_dec/2^INPUT\_WIDTH);

data\_lut\_i = mod(data\_lut\_96\_dec,2^INPUT\_WIDTH);

data\_lut\_q = data\_lut\_q-(data\_lut\_q>=2^(INPUT\_WIDTH-1))\*2^INPUT\_WIDTH;

data\_lut\_i = data\_lut\_i-(data\_lut\_i>=2^(INPUT\_WIDTH-1))\*2^INPUT\_WIDTH;

% input\_data

data\_qi = textread('./src\_0506/FPGA\_tr\_2.txt','%s'); %¸ßiµÍq

[doutq, douti] = input\_signal\_qi(INPUT\_WIDTH, 1, data\_qi);

freq\_domain\_plot('input\\_data',(douti+1j\*doutq)/2^8,737.28);

% ÆæÅ¼ºÏÂ·

data\_even\_i = douti(1:2:end-1);

data\_odd\_i = douti(2:2:end);

data\_i\_temp1 = upsample(data\_even\_i,2);

data\_i\_temp2 = upsample(data\_odd\_i,2);

data\_i\_temp3 = [0 data\_i\_temp2(1:end-1)];

data\_i = data\_i\_temp1+data\_i\_temp3;

data\_even\_q = doutq(1:2:end-1);

data\_odd\_q = doutq(2:2:end);

data\_q\_temp1 = upsample(data\_even\_q,2);

data\_q\_temp2 = upsample(data\_odd\_q,2);

data\_q\_temp3 = [0 data\_q\_temp2(1:end-1)];

data\_q = data\_q\_temp1+data\_q\_temp3;

% data\_qi\_eo = textread('./src\_0506/FPGA\_tr\_2.txt','%s'); %¸ßiµÍq

% [data\_i\_1, data\_q\_1] = input\_signal\_qi(INPUT\_WIDTH, 1, data\_qi);

%% LUT\_coef

LUT\_COEF\_Q = reshape(data\_lut\_q(1:43008),512,LUT\_N);

LUT\_COEF\_I = reshape(data\_lut\_i(1:43008),512,LUT\_N);

%% cordic

[amp\_cordic] = fa\_mag\_compute( data\_i,data\_q );

%% amp2lut

amp = amp\_cordic;

amp\_cast = floor(amp/2^4);

data\_lut\_i\_signed = data\_lut\_i-(data\_lut\_i>=2^(INPUT\_WIDTH-1))\*2^(INPUT\_WIDTH);

amp2lut\_lut = mod(data\_lut\_96\_dec,2^9);

amp2lut\_amp = amp2lut\_lut(amp\_cast+43008+1);

%%

% m=18;n=17;

coef\_q\_lut= zeros(Depth,Depth);

coef\_i\_lut= zeros(Depth,Depth);

coef\_q\_lut\_dual= zeros(Depth,Depth);

coef\_i\_lut\_dual= zeros(Depth,Depth);

dual\_mode\_coef\_i = zeros(Depth,Depth);

dual\_mode\_coef\_q = zeros(Depth,Depth);

coef\_sum\_i = zeros(Depth,1);

coef\_sum\_q = zeros(Depth,1);

coef\_sum = zeros(Depth,1);

data\_complex\_mult = zeros(Depth,1);

dout\_coef\_sum\_i = zeros(Depth,1);

dout\_coef\_sum\_q = zeros(Depth,1);

data\_out = zeros(length(data\_i),1);

data\_out\_i = zeros(length(data\_i),1);

data\_out\_q = zeros(length(data\_i),1);

for pp =1:1:length(data\_i)

for m = Depth:-1:1

for n = Depth:-1:1

% µ±Ç°Êý¾Ý²élut±íÖ¸Õë

mag\_index = pp-Depth+m + LUT\_AN(m,n);

% ÅÐ¶Ïµ±Ç°Î»ÖÃÁÐÏòÁ¿£¬LUT\_LUTÖµÊÇ·ñÎª0

if(LUT\_LUT(m,n) > 0)

lut\_index = LUT\_LUT(m,n);

else

lut\_index = 1;

end

% ÓÃÖ¸Õë²é±íÄ£Öµ

if ((0 < mag\_index)&&(mag\_index <= length(data\_i)))

amp2lut\_coef\_q = LUT\_COEF\_Q(amp2lut\_amp(mag\_index)+1,lut\_index);

amp2lut\_coef\_i = LUT\_COEF\_I(amp2lut\_amp(mag\_index)+1,lut\_index);

else

amp2lut\_coef\_q = 0;

amp2lut\_coef\_i = 0;

end

% ²é´«Í³lut±í

coef\_q\_lut(m,n) = (LUT\_LUT(m,n)>0)\*amp2lut\_coef\_q;

coef\_i\_lut(m,n) = (LUT\_LUT(m,n)>0)\*amp2lut\_coef\_i;

% Ë«Ä£Öµµ±Ç°Êý¾ÝÖ¸Õë

dual\_mag\_index = pp-Depth+m;

% ²éË«Ä£Öµlut±íÁÐÏòÁ¿

if(LUT\_DUAL\_MODE(m,n) > 0)

lut\_dual\_index = LUT\_DUAL\_MODE(m,n);

else

lut\_dual\_index = 1;

end

% ÓÃË«Ä£ÖµÖ¸Õë²éË«Ä£Öµ±í

if ((0 < dual\_mag\_index)&&(dual\_mag\_index <= length(data\_i)))

dual\_amp2lut\_coef\_q = LUT\_COEF\_Q(amp2lut\_amp(dual\_mag\_index)+1,lut\_dual\_index);

dual\_amp2lut\_coef\_i = LUT\_COEF\_I(amp2lut\_amp(dual\_mag\_index)+1,lut\_dual\_index);

else

dual\_amp2lut\_coef\_q = 0;

dual\_amp2lut\_coef\_i = 0;

end

coef\_q\_lut\_dual(m,n) = (LUT\_DUAL\_MODE(m,n)>0)\*dual\_amp2lut\_coef\_q;

coef\_i\_lut\_dual(m,n) = (LUT\_DUAL\_MODE(m,n)>0)\*dual\_amp2lut\_coef\_i;

% if (pp == 12 && m == 11 && n == 1)

% keyboard

% end

% Ë«Ä£Öµµ±Ç°Êý¾ÝÖ¸Õë

dual\_full\_mag\_index = pp-Depth+m+LUT\_BN(m,n);

% È«¾«¶ÈÄ£Öµ³ËË«Ä£ÖµÏµÊý

if ((0 < dual\_full\_mag\_index)&&(dual\_full\_mag\_index <= length(data\_i)))

dual\_mode\_coef\_q(m,n) = floor(amp(dual\_full\_mag\_index).\* (coef\_q\_lut\_dual(m,n)./1024));

dual\_mode\_coef\_i(m,n) = floor(amp(dual\_full\_mag\_index).\* (coef\_i\_lut\_dual(m,n)./1024));

else

dual\_mode\_coef\_q(m,n) = 0;

dual\_mode\_coef\_i(m,n) = 0;

end

end

%¼ÆËã¸÷Éî¶ÈÄÚµÄÏµÊýºÍ

coef\_sum\_i(m,1) = sum(coef\_i\_lut(m,:)) + sum(dual\_mode\_coef\_i(m,:));

coef\_sum\_q(m,1) = sum(coef\_q\_lut(m,:)) + sum(dual\_mode\_coef\_q(m,:));

%Êý¾Ý·À±¥ºÍÒç³ö

index\_i = find(coef\_sum\_i >= 2^(18-1));

coef\_sum\_i(index\_i) = 2^(18-1)-1;

index\_ii = find(coef\_sum\_i <= -2^(18-1));

coef\_sum\_i(index\_ii) = (-2^(18-1));

%

index\_q = find(coef\_sum\_q >= 2^(18-1));

coef\_sum\_q(index\_q) = 2^(18-1)-1;

index\_qq = find(coef\_sum\_q <= -2^(18-1));

coef\_sum\_q(index\_qq) = (-2^(18-1));

% µ±Ç°¼ÇÒäÉî¶È¶ÔÓ¦µÄÊý¾ÝÖ¸Õë

data\_index = pp-Depth+m;

% ÏµÊý±äÎª¸´Êý

coef\_sum(m,1) = coef\_sum\_i(m) + j\* coef\_sum\_q(m);

% ¸´³Ë

if((0 < data\_index)&&(data\_index <= length(data\_i)))

data\_complex\_mult(m,1) = (data\_i(data\_index)+ j\* data\_q(data\_index))\*coef\_sum(m,1);

else

data\_complex\_mult(m,1) = 0;

end

if (pp == 12 && m == 1 && n == 1)

keyboard

end

end

% ¸÷¼ÇÒäÉî¶ÈÏà¼Ó

data\_out\_pp = sum(data\_complex\_mult);

% Êý¾Ý½ØÎ»

data\_out\_pp\_cut = floor(data\_out\_pp/2^13);

data\_out\_pp\_cut\_i = real(data\_out\_pp\_cut);

data\_out\_pp\_cut\_q = imag(data\_out\_pp\_cut);

%Êý¾Ý·À±¥ºÍÒç³ö

index\_i = find(data\_out\_pp\_cut\_i >= 2^(16-1));

data\_out\_pp\_cut\_i(index\_i) = 2^(16-1)-1;

index\_ii = find(data\_out\_pp\_cut\_i <= -2^(16-1));

data\_out\_pp\_cut\_i(index\_ii) = (-2^(16-1));

%

index\_q = find(data\_out\_pp\_cut\_q >= 2^(16-1));

data\_out\_pp\_cut\_q(index\_q) = 2^(16-1)-1;

index\_qq = find(data\_out\_pp\_cut\_q <= -2^(16-1));

data\_out\_pp\_cut\_q(index\_qq) = (-2^(16-1));

%%

data\_out\_i(pp,1) = data\_out\_pp\_cut\_i.';

data\_out\_q(pp,1) = data\_out\_pp\_cut\_q.';

end

%%

fs = 737.28;

data\_A = data\_out\_i(1:end)+data\_out\_q(1:end)\*j;

y1 = 20\*log10(fftshift(abs(fft(data\_A)/length(data\_A)/2^15)));

xa = linspace(-fs/2,fs/2,length(data\_A));

figure(3)

plot(xa',y1);grid on;

title('data\\_A')

%% SRC\_from\_LT

if (COMPARE == 1)

data\_qi\_even = textread('./src\_0506/FPGA\_tr\_DPD\_odd.txt','%s'); %¸ßiµÍq

data\_qi\_odd = textread('./src\_0506/FPGA\_tr\_DPD\_even.txt','%s'); %¸ßiµÍq

[data\_out\_lt\_even\_q, data\_out\_lt\_even\_i] = input\_signal\_qi(INPUT\_WIDTH, 1, data\_qi\_even(1:end-1));

[data\_out\_lt\_odd\_q, data\_out\_lt\_odd\_i] = input\_signal\_qi(INPUT\_WIDTH, 1, data\_qi\_odd);

% ÆæÅ¼ºÏÂ·

data\_i\_temp1 = upsample(data\_out\_lt\_even\_i,2);

data\_i\_temp2 = upsample(data\_out\_lt\_odd\_i,2);

data\_i\_temp3 = [0 data\_i\_temp2(1:end-1)];

data\_i\_lt = data\_i\_temp1.'+data\_i\_temp3.';

data\_q\_temp1 = upsample(data\_out\_lt\_even\_q,2);

data\_q\_temp2 = upsample(data\_out\_lt\_odd\_q,2);

data\_q\_temp3 = [0 data\_q\_temp2(1:end-1)];

data\_q\_lt = data\_q\_temp1.'+data\_q\_temp3.';

data\_LT = data\_i\_lt + 1i\*data\_q\_lt;

fs = 737.28;

y1 = 20\*log10(fftshift(abs(fft(data\_LT)/length(data\_LT)/2^15)));

xa = linspace(-fs/2,fs/2,length(data\_LT));

figure(4)

plot(xa',y1);grid on;

title('SRC\\_from\\_LT');

data\_after\_PA=rapp(data\_i\_lt)+1j\*rapp(data\_q\_lt);

y1 = 20\*log10(fftshift(abs(fft(data\_after\_PA)/length(data\_after\_PA)/2^15)));

xa = linspace(-fs/2,fs/2,length(data\_after\_PA));

figure(5)

plot(xa',y1);grid on;

title('SRC\\_from\\_LT');

else

end

[e,f]=xcorr(data\_out\_i,data\_i\_lt);

figure(1)

plot(f,e);

PA\_DPD.m:

INPUT\_WIDTH = 16;

data\_qi = textread('D:/study/MATLAB/matlab\_code/fdd\_dpd\_matlab/src\_0506/FPGA\_tr\_2.txt','%s'); %¸ßiµÍq

[doutq, douti] = input\_signal\_qi(INPUT\_WIDTH, 1, data\_qi);

freq\_domain\_plot('input\\_data',(douti+1j\*doutq)/2^8,737.28);

%Éú³ÉPAµÄvolterra¼¶ÊýÄ£ÐÍ

x=linspace(0,1,100);

y=rapp(x);

H=[x',(x.\*abs(x))',(x.\*(abs(x)).^2)',(x.\*(abs(x)).^3)'];

w=(((H')\*H)^(-1))\*(H')\*(y');

%compare between the real PA model and the volterra model

y1=0.903981054880251\*x+0.753885617861268\*(x.\*abs(x))-1.55820655404427\*(x.\*(abs(x)).^2)+0.393753321920699\*(x.\*(abs(x)).^3);

subplot(121)

plot(x,y);

subplot(122)

plot(x,y1)

%generating the volterra model of the related DPD model

y\_reverse=x./y;

y\_reverse(1)=1;

H1=[ones(100,1),x',(x.\*abs(x))',(x.\*(abs(x)).^2)',(x.\*(abs(x)).^3)'];

w\_reverse=(((H1')\*H1)^(-1))\*(H1')\*(y\_reverse');

y\_reverse1=0.999999999999846+(3.11439762867849e-12)\*x-(8.23341395062016e-12)\*(x.\*abs(x))+(1.59801061272447e-11)\*(x.\*(abs(x)).^2)+0.999999999998249\*(x.\*(abs(x)).^3);

%y\_reverse1=0.999999999999846+(3.11439762867849\*10^(-12))\*x-(8.23341395062016\*10^(-12))\*(x.\*abs(x))+(59801061272447\*10^(-11))\*(x.\*(abs(x)).^2)+0.999999999998249\*(x.\*(abs(x)).^3);

figure

subplot(121)

plot(x,y\_reverse);

subplot(122)

plot(x,y\_reverse1)

%sim dpd+pa

% u1=u/2;

u=0.999999999999846+(3.11439762867849e-12)\*x-(8.23341395062016e-12)\*(x.\*abs(x))+(1.59801061272447e-11)\*(x.\*(abs(x)).^2)+0.999999999998249\*(x.\*(abs(x)).^3);%after dpd

z=0.903981054880251\*x+0.753885617861268\*(x.\*abs(u1))-1.55820655404427\*(x.\*(abs(x)).^2)+0.393753321920699\*(x.\*(abs(x)).^3);

figure

subplot(131)

plot(x,x)

title('input')

subplot(132)

plot(x,u)

title('after dpd')

subplot(133)

plot(x,z.\*u)

title('after dpd and pa')

%using the volterra model to sim PA model

%x---dpd--->u----pa--->y

power=data\_i.^2+data\_q.^2;

amp\_max=sqrt(mean(power));

data\_i\_scale=abs(data\_i/amp\_max);

data\_q\_scale=abs(data\_q/amp\_max);

u\_i=0.999999999999846+(3.11439762867849e-12)\*data\_i\_scale-(8.23341395062016e-12)\*(data\_i\_scale.\*abs(data\_i\_scale))+(1.59801061272447e-11)\*(data\_i\_scale.\*(abs(data\_i\_scale)).^2)+0.999999999998249\*(data\_i\_scale.\*(abs(data\_i\_scale)).^3);%after dpd

u\_q=0.999999999999846+(3.11439762867849e-12)\*data\_q\_scale-(8.23341395062016e-12)\*(data\_q\_scale.\*abs(data\_q\_scale))+(1.59801061272447e-11)\*(data\_q\_scale.\*(abs(data\_q\_scale)).^2)+0.999999999998249\*(data\_q\_scale.\*(abs(data\_q\_scale)).^3);%after dpd

z\_i=0.903981054880251\*data\_i\_scale+0.753885617861268\*(data\_i\_scale.\*abs(data\_i\_scale))-1.55820655404427\*(data\_i\_scale.\*(abs(data\_i\_scale)).^2)+0.393753321920699\*(data\_i\_scale.\*(abs(data\_i\_scale)).^3);

z\_q=0.903981054880251\*data\_q\_scale+0.753885617861268\*(data\_q\_scale.\*abs(data\_q\_scale))-1.55820655404427\*(data\_q\_scale.\*(abs(data\_q\_scale)).^2)+0.393753321920699\*(data\_q\_scale.\*(abs(data\_q\_scale)).^3);

freq\_domain\_plot('input\\_ori\\_data',(data\_i+1j\*data\_q)/2^8,122.88)