

# Institutionen för systemteknik

## Department of Electrical Engineering

**Examensarbete**

### **Linearization of Power Amplifier using Digital Predistortion, Implementation on FPGA**

Examensarbete utfört i Elektroniksystem  
vid Tekniska högskolan vid Linköpings universitet  
av

**Erik Andersson och Christian Olsson**

LiTH-ISY-EX-14/4803-SE

Linköping 2014



**Linköpings universitet**  
**TEKNISKA HÖGSKOLAN**



# **Linearization of Power Amplifier using Digital Predistortion, Implementation on FPGA**

Examensarbete utfört i Elektroniksystem  
vid Tekniska högskolan vid Linköpings universitet  
av

**Erik Andersson och Christian Olsson**


LiTH-ISY-EX-14/4803-SE

Handledare: **Kent Palmkvist**  
ISY, Linköpings universitet  
**Hampus Thorell**  
FOI

Examinator: **Håkan Johansson**  
ISY, Linköpings universitet

Linköping, 16 november 2014



	<b>Avdelning, Institution</b> Division, Department  Avdelningen för Elektroniksystem Department of Electrical Engineering SE-581 83 Linköping	<b>Datum</b> Date  2014-11-16
<b>Språk</b> Language  <input type="checkbox"/> Svenska/Swedish <input checked="" type="checkbox"/> Engelska/English  <input type="checkbox"/> _____	<b>Rapporttyp</b> Report category  <input type="checkbox"/> Licentiatavhandling <input checked="" type="checkbox"/> Examensarbete <input type="checkbox"/> C-uppsats <input type="checkbox"/> D-uppsats <input type="checkbox"/> Övrig rapport <input type="checkbox"/> _____	<b>ISBN</b> _____ <b>ISRN</b> LiTH-ISY-EX-14/4803-SE <b>Serietitel och serienummer</b> <b>ISSN</b> Title of series, numbering              _____
<b>URL för elektronisk version</b>  <a href="http://www.ep.liu.se">http://www.ep.liu.se</a>		
<b>Titel</b> Title                      Linearization of Power Amplifier using Digital Predistortion, Implementation on FPGA   <b>Författare</b> Erik Andersson och Christian Olsson Author		
<b>Sammanfattning</b> Abstract  <p>The purpose of this thesis is to linearize a power amplifier using digital predistortion. A power amplifier is a nonlinear system, meaning that when fed with a pure input signal the output will be distorted. The idea behind digital predistortion is to distort the signal before feeding it to the power amplifier. The combined distortions from the predistorter and the power amplifier will then ideally cancel each other. In this thesis, two different approaches are investigated and implemented on an FPGA. The first approach uses a nonlinear model that tries to cancel out the nonlinearities of the power amplifier. The second approach is model-free and instead makes use of a look-up table that maps the input to a distorted output. Both approaches are made adaptive so that the parameters are continuously updated using adaptive algorithms.</p> <p>First the two approaches are simulated and tested thoroughly with different parameters and with a power amplifier model extracted from the real amplifier. The results are shown satisfactory in the simulations, giving good linearization for both the model and the model-free technique.</p> <p>The two techniques are then implemented on an FPGA and tested on the power amplifier. Even though the results are not as well as in the simulations, the system gets more linear for both the approaches. The results vary widely due to different circumstances such as input frequency and power. Typically, the distortions can be attenuated with around 10 dB. When comparing the two techniques with each other, the model-free method shows slightly better results.</p>		
<b>Nyckelord</b> Keywords              digital predistortion, power amplifier, nonlinear models, FPGA, linearization, jamming		



## Abstract

The purpose of this thesis is to linearize a power amplifier using digital predistortion. A power amplifier is a nonlinear system, meaning that when fed with a pure input signal the output will be distorted. The idea behind digital predistortion is to distort the signal before feeding it to the power amplifier. The combined distortions from the predistorter and the power amplifier will then ideally cancel each other. In this thesis, two different approaches are investigated and implemented on an FPGA. The first approach uses a nonlinear model that tries to cancel out the nonlinearities of the power amplifier. The second approach is model-free and instead makes use of a look-up table that maps the input to a distorted output. Both approaches are made adaptive so that the parameters are continuously updated using adaptive algorithms.

First the two approaches are simulated and tested thoroughly with different parameters and with a power amplifier model extracted from the real amplifier. The results are shown satisfactory in the simulations, giving good linearization for both the model and the model-free technique.

The two techniques are then implemented on an FPGA and tested on the power amplifier. Even though the results are not as well as in the simulations, the system gets more linear for both the approaches. The results vary widely due to different circumstances such as input frequency and power. Typically, the distortions can be attenuated with around 10 dB. When comparing the two techniques with each other, the model-free method shows slightly better results.





## Acknowledgments

First of all we would like to thank FOI and especially Hampus Thorell for giving us the opportunity to do this thesis. We would also like to thank many others on FOI for all the help during this work. A big thanks also to our examiner Prof. Håkan Johansson and our supervisor Dr. Kent Palmkvist.

Finally we would like to thank all our friends and family for all the support during this time.

*Linköping, November 2014  
Erik Andersson and Christian Olsson*



---

# Contents

<b>Notation</b>	<b>xi</b>
<b>1 Introduction</b>	<b>1</b>
1.1 Background . . . . .	1
1.2 Purpose . . . . .	2
1.3 Approach . . . . .	3
1.4 Requirements . . . . .	3
1.5 Tools and Hardware . . . . .	4
1.5.1 Software . . . . .	4
1.5.2 Hardware . . . . .	4
1.6 Outline . . . . .	5
<b>2 Power Amplifier</b>	<b>7</b>
2.1 Gain . . . . .	7
2.2 Efficiency . . . . .	8
2.3 Linearity . . . . .	8
2.3.1 Decibels Relative to Carrier . . . . .	8
2.3.2 Gain Compression . . . . .	8
2.3.3 Adjacent Channel Power Ratio . . . . .	9
2.4 Power Amplifier Memory Effects . . . . .	10
2.5 Classification of Power Amplifiers . . . . .	10
2.5.1 Class A . . . . .	11
2.5.2 Class B . . . . .	11
2.5.3 Class AB . . . . .	12
2.5.4 Class C . . . . .	12
<b>3 Modeling of Power Amplifier</b>	<b>15</b>
3.1 Nonlinearities . . . . .	16
3.1.1 Distortion . . . . .	16
3.1.2 Properties . . . . .	17
3.2 Memoryless Models . . . . .	18
3.2.1 Saleh Model . . . . .	19
3.2.2 Memoryless Polynomial Model . . . . .	19

3.2.3	Other Memoryless Models . . . . .	19
3.3	Memory Models . . . . .	19
3.3.1	Volterra Series . . . . .	20
3.3.2	Hammerstein Model . . . . .	20
3.3.3	Wiener Model . . . . .	21
3.3.4	Wiener-Hammerstein Model . . . . .	21
3.3.5	Nonlinear Moving Average Model . . . . .	22
3.3.6	Nonlinear Auto-Regressive Moving Average Model . . . . .	22
3.3.7	Other Memory Models . . . . .	23
3.4	Parameter Extraction . . . . .	24
3.4.1	Least Squares Method . . . . .	24
3.4.2	Adaptive Least Squares Methods . . . . .	24
3.5	Conclusions . . . . .	26
<b>4</b>	<b>Linearization Techniques</b>	<b>27</b>
4.1	Feedback . . . . .	27
4.2	Feedforward . . . . .	28
4.3	Predistortion . . . . .	29
4.3.1	Analog Predistortion . . . . .	29
4.3.2	Digital Predistortion . . . . .	30
4.4	Conclusions . . . . .	32
<b>5</b>	<b>Measurements</b>	<b>33</b>
5.1	Measurement Setup . . . . .	33
5.2	Measurement Results . . . . .	34
<b>6</b>	<b>Simulations</b>	<b>37</b>
6.1	Power Amplifier Modeling . . . . .	37
6.2	Motivation and Decisions . . . . .	39
6.2.1	Model Implementation . . . . .	39
6.2.2	Model-free Implementation . . . . .	40
6.2.3	Input Signal . . . . .	40
6.3	NMA Simulation . . . . .	40
6.3.1	Results . . . . .	41
6.4	LUT Simulation . . . . .	44
6.4.1	Method . . . . .	44
6.4.2	Results . . . . .	46
<b>7</b>	<b>Implementation</b>	<b>51</b>
7.1	Base Implementation . . . . .	51
7.1.1	Direct Digital Synthesizer . . . . .	51
7.1.2	System Overview . . . . .	52
7.1.3	Hardware Resources . . . . .	52
7.2	Adaptive Architecture . . . . .	53
7.2.1	Calibration Sensitivity . . . . .	54
7.3	NMA Implementation . . . . .	54
7.3.1	Horner's Method . . . . .	54

---

7.3.2	Number Representation . . . . .	55
7.3.3	DPD . . . . .	55
7.3.4	APD . . . . .	57
7.3.5	Results . . . . .	57
7.3.6	Hardware Resources . . . . .	61
7.4	LUT Implementation . . . . .	61
7.4.1	DPD . . . . .	61
7.4.2	APD . . . . .	62
7.4.3	Results . . . . .	64
7.4.4	Hardware Resources . . . . .	67
<b>8</b>	<b>Conclusions and Future Work</b>	<b>69</b>
8.1	Conclusions . . . . .	69
8.1.1	Implementation . . . . .	70
8.2	Future Work . . . . .	71
	<b>Bibliography</b>	<b>73</b>



---

# Notation

## ABBREVIATIONS A-L

Abbreviation	Meaning
ACPR	Adjacent channel power ratio
ADC	Analog to digital converter
APD	Adaptive predistortion
AM	Amplitude modulation
DAC	Digital to analog converter
DC	Direct current
DDS	Direct digital synthesis
DFT	Discrete Fourier transform
DLA	Direct learning architecture
DPD	Digital predistortion
DSP	Digital signal processing
FFT	Fast Fourier transform
FIR	Finite impulse response
FOI	Totalförsvarets forskningsinstitut (Swedish defence re- search agency)
FPGA	Field programmable gate array
GSM	Global system for mobile communications
HM	Harmonic
ILA	Indirect learning architecture
IM	Intermodulation
LMS	Least mean square
LTE	Long term evolution
LTI	Linear time invariant
LUT	Look-up table

**ABBREVIATIONS N-Z**

<b>Abbreviation</b>	<b>Meaning</b>
NARMA	Nonlinear auto regressive moving average
NMA	Nonlinear moving average
NMSE	Normalized mean square error
PA	Power amplifier
PAE	Power added efficiency
PM	Phase modulation
RF	Radio frequency
RLS	Recursive least square
VHDL	VHSIC (Very high speed integrated circuit) high description language



# 1

---

## Introduction

A key component when transmitting a radio frequency signal (RF-signal) is the power amplifier (PA). It is responsible to amplify the signal to the wanted power level. The higher power level the further the signal can travel in the particular medium (e.g. air). A problem with the analog PA is its nonlinear behavior. When driven harder and towards saturation the nonlinear effects increase. The result of the nonlinear effects is that the signal will be distorted with harmonics (HM) and intermodulation (IM) products. This is called spectral regrowth and for anyone transmitting in these frequency bands, these signals will be perceived as distortion. The purpose of this thesis is to investigate the nonlinear behavior of a PA. Different linearization methods will be analyzed and implemented to try to attenuate these distortions. In Fig. 1.1 the HM and IM distortion can be seen for an arbitrary nonlinear system. A desired linear output would have been only the two tones at  $f_1$  and  $f_2$  in the so called fundamental zone.

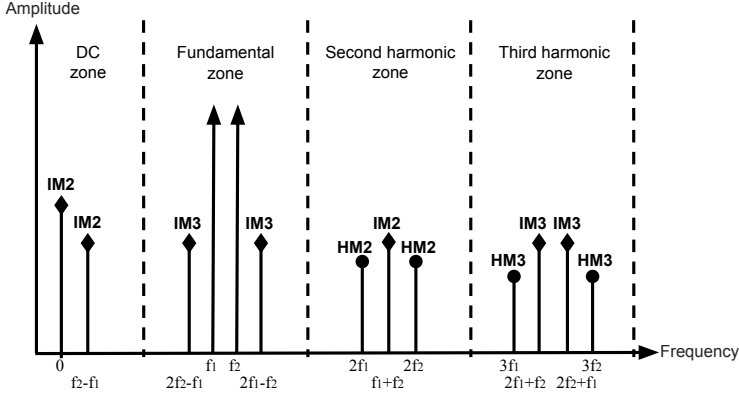
### 1.1 Background

This master thesis is carried out at Totalförsvarets Forskningsinstitut (Swedish Defence Research Agency), from here on referred to as FOI. Commercially a PA is mostly used within telecommunication in the base stations to help transmit signals according to the GSM<sup>1</sup> and LTE-advanced<sup>2</sup> standards. Today it is common that the signals have a wide bandwidth and several nearby carrier frequencies i.e. channels. This is implemented to have a higher data rate. This also gives more problems with the IM products due to the distortion within the fundamental

---

<sup>1</sup>Global system for mobile communications. Technology used in the second generation cellular phone system.

<sup>2</sup>Long term evolution advanced. Technology used in the fourth generation cellular phone system.



**Figure 1.1:** A two tone test applied to an arbitrary nonlinear system. The distortions from HM and IM are visible. A desired linear output would have been only the two inputted frequencies  $f_1$  and  $f_2$  in the fundamental zone.

zone. One does not want to interfere with nearby channels because the transmitting signal in these channels will then have higher bit error rate<sup>3</sup> at the receiver. The HM distortion and other distortion outside the fundamental zone is easily taken care of by putting an appropriate analog bandpass filter around the fundamental zone.

But when it comes to the intention of this thesis, the application is meant for jamming. Here the intention is not to communicate but more to interfere on a wide range of selected frequencies to unable communication for someone else. Jamming can for example be used to prevent terror threats were IEDs<sup>4</sup> are detonated through cellphones. In that case jamming would prevent the signal and therefore avoid a detonation of the bomb. More related to this thesis is jamming of enemy radio. Desired qualities for this is to have a wide range of selective frequencies to be able to jam. Also the ability to quickly jump between frequencies using frequency hopping to have a strategic way of jamming enemy radio is desired. For all these applications it is not desired to transmit on frequencies not intended to, therefore it is highly desired to linearize the PA and remove the nonlinear effects. For example the distortions could end up in civil frequencies or frequency bands that is used by oneself. There are regulations and standards describing the amount of power that is allowed to be spread to adjacent frequencies.

## 1.2 Purpose

The goal of this thesis is to construct a combined linear system that includes a digital system and the PA, used for jamming. Also achieved from a more linear

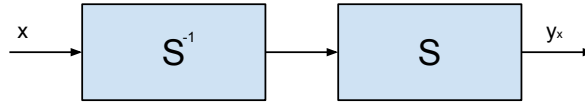
<sup>3</sup>A proportional measurement of how many bits that have been altered due to interference.

<sup>4</sup>Improvised explosive device. A homemade bomb.

system is the efficiency, this is desired since the power consumption will be lower and the lifetime of a mobile system will be prolonged.

## 1.3 Approach

The PA is as mentioned earlier a nonlinear system, here called  $S$ . The goal is to implement digital predistortion (DPD) and construct a predistorter which is by itself a nonlinear system. If this predistorter has the exact inversed behavior compared to  $S$  it would be a nonlinear system  $S^{-1}$ . Knowledge of nonlinear systems and modeling of such systems are then needed to construct a predistorter. Cascading these two systems would ideally become a linear system, see Fig. 1.2. The distortions from the two systems will then combined cancel each other.



**Figure 1.2:** A system  $S$  with a cascaded exact predistorter  $S^{-1}$  would give a combined linear system.

The PA is a complex system and finding an inverse to a nonlinear system is not always possible. When it comes to the implementation, a more complex system usually results in more computation power and hardware. The goal is to find the least complex predistorter that will find an approximate inverse that satisfies the linearity requirements.

## 1.4 Requirements

The operation bandwidth of the used PA is 20-520 MHz. However, at FOI it is used between 30-520 MHz. But the ADC has a maximum clock frequency of 409.6 MHz. This means that according to the Nyquist criterion the digital base-band is limited to 204.8 MHz if the sampled signal should be free from aliasing. The final combined system should be able to work over the range 30-204.8 MHz. The desired linearity requirement is to have no nonlinear terms higher than -40 dBc. It is also highly desired to have a maximum output power of the fundamental tone which directly conflicts with the linearity. The maximum output power from the used DAC is 3 dBm, that corresponds to a full voltage swing. However, the PA already has a strongly saturated output when fed with an input of 0 dBm, giving the same fundamental power level at the output. This is a trade-off to be considered. The performance of a PA depends on the ambient temperature which vary over time, this should also be considered in the linearization process.

In this thesis the application is meant for jamming which means that no modulation of the signal is required. Therefore the focus will not be on complex signals but instead on real valued signals only.

## 1.5 Tools and Hardware

Several softwares have been used during the process of this thesis. Also different hardware components were needed to build up the total system.

### 1.5.1 Software

#### **MATLAB**

MATLAB is a program created by Mathworks Inc that uses a high-level language for numerical computation. This program was used to develop models and algorithms that was later to be mapped to the FPGA. MATLAB was also used to analyze measured and simulated data.

#### **Xilinx ISE and iSim**

Integrated Software Environment (ISE) is a program created by Xilinx for synthesis and analysis of hardware description language (HDL) designs. The syntax of this type of programming languages has a purpose of capturing the parallelism of the desired hardware design. Through this program the code can be compiled and the design can be synthesized which generates a bit file that can be loaded into an FPGA. Two common hardware description languages are Verilog and VHDL, where the latter is used for this thesis. Within this software environment a simulator called iSim is integrated that makes it able to simulate the design. The result from the simulations was then used for verification as well in MATLAB for analysis.

#### **Xilinx ChipScope**

ChipScope is a software tool created by Xilinx that is used to debug and verify digital designs after it has been synthesized. By adding Chipscope modules in the digital design this program makes it possible to record signals and study them much like a logic analyzer. This tool is very helpful to verify if simulation and measurement results match. This makes it possible to debug and exclude problems like hardware malfunction or timing constraints not being met. Measurement results from ChipScope were also used for analysis.

### 1.5.2 Hardware

#### **FPGA**

FPGA stands for Field Programmable Gate Array and is the key component in the system where the digital design is to be implemented. An FPGA is an integrated circuit that is very appealing since it can be reconfigurable after manufacturing. The FPGA mainly consists of slices that builds up look up tables (LUTs) and flip-flop registers. A LUT is basically a read-only memory and since the LUT and the interconnections are configurable, any form of combinational logic function can be designed. Assembling them together with the registers a sequential logic net can be implemented. It is nowadays also common with sets of dedicated specialized hardware like multipliers, multiply and accumulate units, memories

and phase locked loops that are often used in digital designs. The used FPGA in this thesis is a Xilinx Virtex 5.

### **MPX**

The MPX is a development board created by FOI. The board contains key components such as the FPGA, a DAC, and an ADC. The routing is designed so pins for input and output are easily obtained as well as a USB interface. It has also other necessary parts such as power network and an internal and external clock distribution.

### **Power Amplifier**

The PA is the system that is aimed to be linearized. The used PA is a solid state circuit of class AB from Empower RF. The modeltype is a BBM2E3KKO (SKU 1094). Further information on power amplifiers can be found in Chapter 2.

## **1.6 Outline**

Chapter 2 of this thesis gives an overview of the theory behind power amplifiers. Chapters 3 and 4 contain general theory regarding nonlinearities and modeling of nonlinear systems such as the PA. They also include information of different linearization techniques like the implemented DPD technique. Chapter 5 shows measurements that have been performed on the used PA. Chapters 6 and 7 contain simulations and implementation of the algorithms for the linearization techniques. In Chapter 8 conclusions and possible future work are presented.



# 2

---

## Power Amplifier

The PA is one of the most important components in an RF transmitter. It is used to amplify the signal before transmitting it to the antenna. The PA is by far the most power-hungry component in the transmitter and it consumes the main part of the total power. Therefore, it is important to try to keep the efficiency high. However, a very efficiently designed PA will have a more nonlinear behavior. That also works the other way around i.e. a very linearly designed amplifier is not very efficient. The trade-off between linearity and efficiency is one of the main problem in todays PAs. What is preferred depends greatly on the application. For example in a cellular phone the efficiency should be kept high to make the battery time longer.

This chapter gives a short introduction to the PA. It is by no means a full description of the PA but enough to understand this thesis.

### 2.1 Gain

One of the most important properties of a PA is its gain. It describes the relation between the input and the output of the amplifier. Usually it is defined as

$$A = \frac{\text{Output}}{\text{Input}}. \quad (2.1)$$

If *Input* and *Output* are voltages, the voltage gain,  $A_V$ , is obtained. Similarly the power gain,  $A_P$ , is obtained if the signals are powers. Usually it is expressed in decibel (dB) as

$$A_P = 10 \log_{10} \left( \frac{P_{out}}{P_{in}} \right) \quad (2.2)$$

where  $P_{out}$  and  $P_{in}$  are the output and input power, respectively.

## 2.2 Efficiency

As mentioned before the efficiency of a PA is important. It is defined as

$$\eta = \frac{P_L}{P_{supp}} \quad (2.3)$$

where  $P_L$  is the average power delivered to the load and  $P_{supp}$  is the average power drained from the supply source. In theory  $\eta$  can reach up to 100% but in practice this is not possible. The power not consumed by the amplification itself, i.e. the power leading up to the full 100%, can be considered as losses.

Another measurement of efficiency is the power added efficiency (PAE), defined as

$$PAE = \frac{P_L - P_{in}}{P_{supp}} \quad (2.4)$$

where  $P_{in}$  is the average input power.

## 2.3 Linearity

The linearity of a PA can be measured in many different ways. Some of them are listed below.

### 2.3.1 Decibels Relative to Carrier

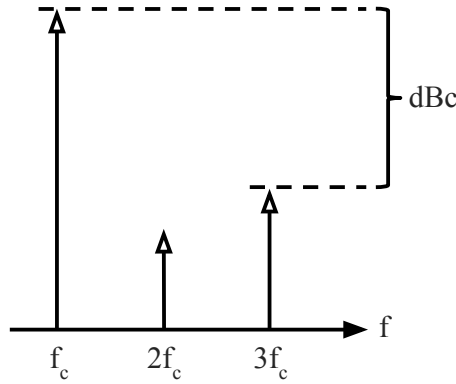
The decibel relative to carrier, or dBc, is a commonly used quantity to measure the linearity. It defines the difference in dB between a signal and a carrier signal. An example is shown in Fig. 2.1 where a carrier is compared with its third harmonic. Since the third harmonic is lower than the carrier the dBc measure will be negative.

### 2.3.2 Gain Compression

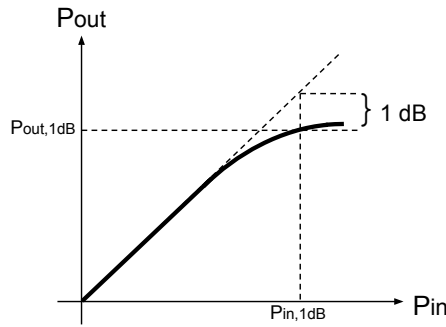
For large inputs the PA will eventually not be able to fully amplify the signal. This is called gain compression and it will give the output a nonlinear behavior. To measure this problem one can use the 1-dB compression point. It is defined as the point where the output power differs from the ideal linear value by 1 dB,



shown in Fig. 2.2. It can be specified by both the input,  $P_{in,1dB}$ , and the output,  $P_{out,1dB}$ , power where this occurs.



**Figure 2.1:** Illustration of the dBc measure.



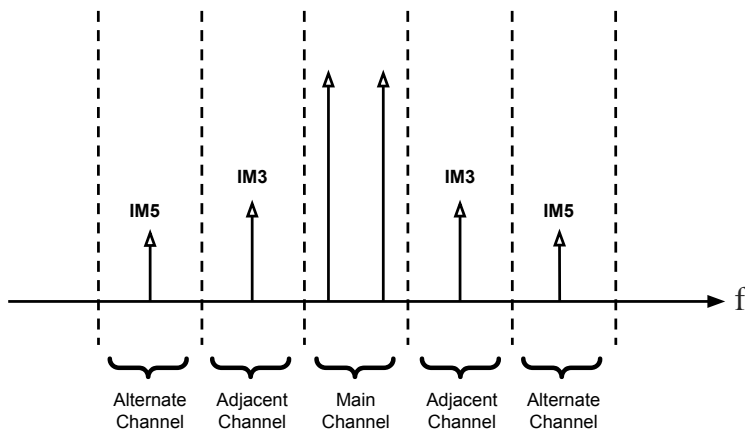
**Figure 2.2:** Gain compression and 1-dB compression point.

### 2.3.3 Adjacent Channel Power Ratio

Another measurement of linearity for modulated signals is the adjacent channel power ratio (ACPR). It is a measurement of how much of the spectrum that has spread to the nearby channel. It is defined as the ratio between the power in the adjacent channel (intermodulation signal) and the power in the main channel. Sometimes the alternate channel power ratio is used. It is basically the same as ACPR but instead the power that is two channels away from the main channel is considered.

To get a rough estimation of the ACPR a two-tone test can be applied. As seen in Fig. 2.3, the modulated signal is then being replaced with tones of the same power level. Due to the nonlinear behavior this will generate a lot of IM products. From the figure it is shown that the 3rd and 5th order IM products will end up inside the adjacent and the alternate channel respectively. The approximate ACPR can then be calculated as the power ratio of one of the IM3 products to the power of

one of the main tones. The alternate channel power ratio can be calculated the same way but using the IM5 instead [Anritsu, 2001].



**Figure 2.3:** Two-tone test for ACPR measure.

## 2.4 Power Amplifier Memory Effects

In some cases the PA does not only introduce amplitude distortion but phase distortion as well. This indicates that the device suffers from memory effects. It basically means that the output of the amplifier is not only dependent on the current input sample but on previous input samples as well.

There are two basic kinds of memory effects called *electrical memory effects* and *electrothermal memory effects*. The electrical memory effects are caused by variable impedances at the DC, fundamental and harmonic band. One source of these variations in impedance comes from the transistors in the bias network. This will generate undesirable signals with the same frequencies as the intermodulation distortion products. The electrothermal memory effects have to do with the fact that the transistors change properties with different temperatures. This will also generate IMD products. Usually this effect is not a significant problem for bandwidth below 1 MHz.

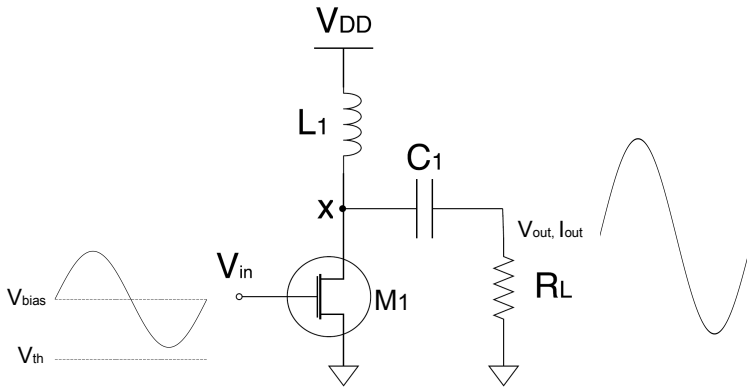
The main issue with memory effects is that it can cause major problems for some linearization techniques, e.g. digital predistortion [Anttila, 2011].

## 2.5 Classification of Power Amplifiers

There are many different kinds of PAs and they are usually divided into different classes depending on their efficiency and linearity. This section summarizes the most common classes of PAs.

### 2.5.1 Class A

The class A amplifier is the most linear of the classes because it operates over the full input and output range. This means, assuming a sinusoidal input, that it amplifies for the whole input cycle, i.e.  $360^\circ$  of the sine wave. Usually this is called the conduction angle. Figure 2.4 shows an example of a class A amplifier. As seen, the bias voltage to the transistor is chosen to be higher than the peak voltage of the input signal. This will make sure that the input voltage is always higher than the threshold voltage making the transistor, M1, always conducting. The efficiency of the class A amplifier is therefore low.



**Figure 2.4:** An example of a class A amplifier.

The maximum efficiency can be calculated by letting node x, in Fig. 2.4, swing between  $2V_{DD}$  and zero. The power delivered to  $R_L$  is then equal to  $(2V_{DD}/2)^2/(2R_{in})$  where  $R_{in}$  is the input impedance. Equation (2.3) and the fact that L1 will drain a constant current of  $V_{DD}/R_{in}$  from the supply gives

$$\eta = \frac{(2V_{DD}/2)^2/(2R_{in})}{V_{DD}^2/R_{in}} = \frac{V_{DD}^2 R_{in}}{2V_{DD}^2 R_{in}} = 50\%. \quad (2.5)$$

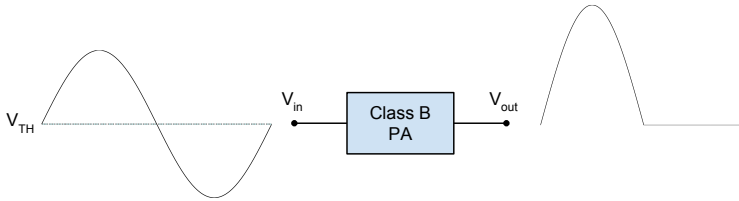
The remaining power is consumed by M1 [Razavi, 1998].

### 2.5.2 Class B

The class B amplifier only conducts for half the input cycle or in other words, only have a conduction angle of  $180^\circ$ . In this case the bias voltage of the input signal is at the threshold voltage of the transistor, see Fig. 2.5. When the input signal goes below  $V_{th}$  the device, M1, will turn off. This decreases the linearity

but increases the efficiency. The maximum efficiency for a class B amplifier is  $\eta = \pi/4 \approx 78.5\%$ .

Usually two class B amplifiers are connected together, making the amplifiers work on half a cycle each. This will make the circuit amplify for the whole input range and in theory this makes a linear amplifier. However, due to mismatches in the transistors this is not the case. The efficiency for this kind of circuit is still 78.5%.



**Figure 2.5:** Input and output relation for the class B amplifier.

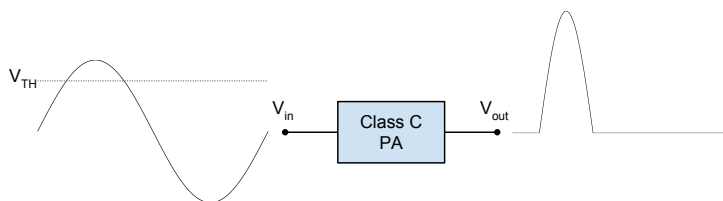
### 2.5.3 Class AB

The class AB amplifier has a conduction angle between the two classes above, i.e. between  $180^\circ$  and  $360^\circ$ . This makes the amplifier more linear than the class B stage but not as linear as the class A design. A class AB amplifier is therefore less efficient than the class B but more efficient than class A.

### 2.5.4 Class C

In a class C amplifier the conduction angle is even lower than for class B. Figure 2.6 shows an example of the input and output of a class C amplifier. Note that the input voltage only exceeds the transistors threshold voltage for a short period of time. Doing this, the amplifier will only output short pulses. There is usually some kind of resonant circuit connected to avoid large harmonic levels at the output.

As the conduction angle approaches zero the efficiency of the class C amplifier will go towards 100%. However, the power delivered to the load will also approach zero. The conduction angle has to be above zero but should still be kept small to have a good efficiency. Since the class C amplifier only conducts for a short period of time, the output current has to be very high to be able to have the same output power as a class A design. This means that the output transistor has to be very wide which is undesirable in modern RF design.



**Figure 2.6:** Input and output relation for the class C amplifier.



# 3

---

## Modeling of Power Amplifier

The PA consists of many electronic components and some of them are inherently nonlinear such as the transistor and the diode. This makes the system itself a nonlinear system. The goal of modeling a nonlinear system in this thesis is to mathematically capture a system's behavior and then use this model when simulating the system. For a given input to the system it is desired that the output from the model mimics the true system output. For the single input and single output system used in this thesis it is then suited to do a behavioral modeling that do not require any knowledge of the internal setup. This is called black-box modeling. In contrast to this we have physical modeling which does require knowledge of the internal components to set up a number of nonlinear equations explaining all voltage and current relations. This gives a very accurate result when done correctly. However, a circuit-level simulation like this is very complex and results in a very long simulation time, making it not essential for this thesis. The main priority is not to have a very accurate model of the PA but only to extract its behavior in simulation to prove the concept of the linearization techniques.

A nonlinear system is a very wide category of systems where no general rules for modeling applies. What is usually done is that knowledge of linear dynamic systems are combined with memoryless nonlinear systems. In this chapter several different concepts of this idea will be presented. All these concepts have a trade-off between complexity and accuracy. If a more mimicked output from the model is desired compared to the true system output, then a mathematical description with more computation can be chosen.

### 3.1 Nonlinearities

For a system to be linear it is said that when fed with a linear combination of inputs the output response is the corresponding linear combination of the individual outputs. This can be written as

$$f(c_1x + c_2y) = f(c_1x) + f(c_2y) = c_1f(x) + c_2f(y) \quad (3.1)$$

where the function  $f$  holds this criteria given that  $x$  and  $y$  are independent variables and  $c_1$  and  $c_2$  are real-valued scalars. Two important properties are needed to be applied for the equation to hold. In the first equality the superposition property is used and for the second equality the homogeneity property is used. Any other system that does not satisfy these properties is a nonlinear system. The easiest form of a nonlinear system is described by a polynomial function

$$y(n) = \alpha_1 \cdot x(n) + \alpha_2 \cdot x^2(n) + \alpha_3 \cdot x^3(n) + \dots + \alpha_P \cdot x^P(n) = \sum_{p=1}^P \alpha_p x^p(n). \quad (3.2)$$

As seen the output is not directly proportional to the input. The first term with the power of one is the linear term. All the other terms with respective weighted coefficient  $\alpha_p$  gives the nonlinear effects. Any function that includes a term that has a power separated from one does not have the previous mentioned properties [Söderkvist and Ahnell, 1994].

#### 3.1.1 Distortion

The nonlinear behavior aimed to be modeled is visible in form of different distortions. In this section these distortions will be mathematically explained. The equation

$$y(t) = \alpha_1 x(t) + \alpha_2 x^2(t) + \alpha_3 x^3(t) \quad (3.3)$$

is a nonlinear model of type (3.2) with order  $P = 3$ . This model is sufficient to extract the desired nonlinear behavior to demonstrate nonlinear distortion.

#### Harmonic Distortion

If the input to (3.3) would contain a single frequency  $x(t) = A \cos(ft)$  only harmonics will be visible at the output i.e. multiples of the input frequency. With help of the double angle formula,

$$2 \cos^2(ft) = 1 + \cos(2ft), \quad (3.4)$$

the following output will be obtained

$$\begin{aligned} y(t) &= \alpha_1 A \cos(ft) + \alpha_2 A^2 \cos^2(ft) + \alpha_3 A^3 \cos^3(ft) \\ &= \alpha_1 A \cos(ft) + \alpha_2 A^2 \left( \frac{1}{2} + \frac{1}{2} \cos(2ft) \right) + \alpha_3 A^3 \left( \frac{3}{4} \cos(ft) + \frac{1}{4} \cos(3ft) \right). \end{aligned} \quad (3.5)$$

As expected there are only multiples of the input frequency  $f$ , spanning from DC



up to the order of the model which is 3. An important observation can be made. The even order coefficient  $\alpha_2$  only affects DC and  $2f$ . Further, the coefficient  $\alpha_3$  only affects  $f$  and  $3f$ . This is a general rule, odd exponents in the model only affects odd HM and even exponents in the model only affects even HM [Teikari, 2008].

### Intermodulation Distortion

If the input to (3.3) contains multiple frequencies as

$x(t) = A(\cos(f_1 t) + \cos(f_2 t))$  then intermodulation products will be visible at the output. The output will then be

$$y(t) = \alpha_1 A(\cos(f_1 t) + \cos(f_2 t)) + \alpha_2 A^2(\cos(f_1 t) + \cos(f_2 t))^2 + \alpha_3 A^3(\cos(f_1 t) + \cos(f_2 t))^3. \quad (3.6)$$

Expanding the nonlinear terms and arranging the result to the respectively weighted coefficient  $\alpha_i$ , the following expression will be found:

$$\begin{aligned} y(t) = & \alpha_1 A(\cos(f_1 t) + \cos(f_2 t)) + \\ & \alpha_2 A^2(1 + \cos((f_2 - f_1)t) + \cos((f_2 + f_1)t) + \frac{1}{2} \cos(2f_1 t) + \frac{1}{2} \cos(2f_2 t)) + \\ & \alpha_3 A^3 \left( \frac{3}{4} \cos((2f_2 - f_1)t) + \frac{3}{4} \cos((2f_1 - f_2)t) + \frac{9}{4} \cos(f_1 t) + \frac{9}{4} \cos(f_2 t) + \right. \\ & \left. \frac{3}{4} \cos((2f_1 + f_2)t) + \frac{3}{4} \cos((2f_2 + f_1)t) + \frac{1}{4} \cos(3f_1 t) + \frac{1}{4} \cos(3f_2 t) \right). \end{aligned} \quad (3.7)$$

What is performed here is called a two-tone test. In Fig. 1.1 a two-tone test for an arbitrary nonlinear system can be seen. The same observations as previously can be made, that the general rule of odd and even exponents only affect in the odd or even zone. This means that implementations that have a bandpass filter over the fundamental zone only need to take account for the odd order effects since they are the only nonlinear effects on the transmitting signal. This is done in most telecommunication implementations since it reduces the model complexity. However, for this thesis the implementation is meant for jamming, so the even order terms will be included as well. An investigation of the effects of the even order terms is presented in [Ding and Zhou, 2004].

### 3.1.2 Properties

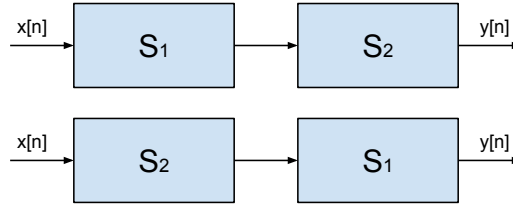
It is important to know the limitations of a system to be able to understand it. Apart from additive (superposition) and homogeneity, a nonlinear system is also not commutative [Jung, 2013]. This means that if we have two cascaded systems as in Fig. 3.1 where at least one of the systems is a nonlinear system, the order of placement is of importance. Giving a simple example, assume that  $S_1 = \alpha x$  and  $S_2 = \beta x^2$ . From these systems the two different cascaded functions  $F_1(x)$  and

$F_2(x)$  are obtained as

$$F_1(x) = S_1(S_2(x)) = \alpha \cdot (\beta x^2) = \alpha \beta x^2 \quad (3.8a)$$

$$F_2(x) = S_2(S_1(x)) = \beta \cdot (\alpha x)^2 = \alpha^2 \beta x^2. \quad (3.8b)$$

This gives that  $F_1(x) \neq F_2(x)$  for all nonzero coefficient values, except  $\alpha = 1$ . This information of commutativity is of relevance when it comes to the linearization techniques of predistorter due to the cascading of systems.



**Figure 3.1:** Two cascaded systems with different order of placement.

A nonlinear system can also be time-invariant. This means that the systems response to a specific input does not depend on absolute time, so the output does not explicitly depend on time. Further, a system can be so called dynamic if the output not only depends on the current input. It is then also dependent on previous inputs, it has a memory or history. More about the origin of memory effects in a PA can be read in Chapter 2. If the system does not depend on previous inputs, just the current input, it is called static. This gives a clear partitioning between two model types, memoryless models and memory models.

## 3.2 Memoryless Models

A memoryless model is only dependent on the current input, there is no history of previous samples that affects the output. There is a one-to-one mapping between current input voltage and output voltage. The memoryless models usually divides the distortion into two parts, the amplitude-to-amplitude (AM/AM) distortion and amplitude-to-phase (AM/PM) distortion. The AM/AM distortion function tries to model the saturated output signal that occurs because of gain compression, see Section 2.3.2. The AM/PM distortion function tries to model the phase shift of the signal. Since the signal phase is dependent on previous samples it is not strictly memoryless, but it is approximated and often then called quasi-memoryless. A strictly memoryless model only models AM/AM distortion.

### 3.2.1 Saleh Model

The Saleh model is a commonly used power amplifier model because of its low parameter complexity, it has four parameter coefficients. It is defined as

$$f_a(|x(k)|) = \frac{\alpha_a |x(k)|}{1 + \beta_a |x(k)|^2} \quad (3.9)$$

$$f_\phi(|x(k)|) = \frac{\alpha_\phi |x(k)|^2}{1 + \beta_\phi |x(k)|^2}. \quad (3.10)$$

The coefficients  $\alpha_a$  and  $\beta_a$  determine the AM/AM distortion and the coefficients  $\alpha_\phi$  and  $\beta_\phi$  determine the AM/PM distortion [Saleh, 1981]. A disadvantage of this model if it was to be implemented on an FPGA is the division, a mathematical operation that is most often tried to avoid, if possible, since it increases the computational complexity.

### 3.2.2 Memoryless Polynomial Model

A polynomial function is maybe the most natural and most used model to describe the PA nonlinear static behavior. The equation

$$y_{Poly}(k) = \sum_{p=1}^P \alpha_p x^p(n) \quad (3.11)$$

is a polynomial function of order  $P$ . The coefficients  $\alpha_p$  determine the distortion. As previously mentioned the model can be simplified by only including odd order terms, if ones interest is only to model IM-products in the fundamental zone. However, this is not an appropriate simplification if the interest is to model HM-products in zones beyond the fundamental.

### 3.2.3 Other Memoryless Models

Other model variants includes Rapp model and Ghorbani model which are similar variations of the Saleh model. There are also several model techniques that expresses the modeled output as a complex Fourier series expansion of the current input signal. A variety of these are called Fourier series model, Bessel-Fourier model, Hetrakul and Taylor model. More about these models can be read in Schreurs et al. [2008].

## 3.3 Memory Models

A memory model is not only dependent on the current input but a certain depth of previous samples will also affect the output, this makes the system dynamic. For accurate modeling some systems also depend on previous output values. The memory effects can be seen as frequency domain variations in the transfer characteristics of the power amplifier. The previously mentioned static models are frequency independent and can represent the characteristics of a PA that is driven

by a narrowband input signal [Schreurs et al., 2008]. For newer modulation techniques, wideband signals are usually used to get a higher data rate. But with wider signals, the memory effects are more apparent. This makes the memory effects a crucial point for a linearization implementation aimed for jamming since a very wide bandwidth is used. In the following sections some of the most common memory models will be discussed to see how they can model these memory effects.

### 3.3.1 Volterra Series

The Volterra model is the most general model which can cover a vast number of possible system states. It is considered as an extension of the Taylor series<sup>1</sup>. In its discrete FIR-form the series can be written as

$$y_{Vol}(k) = \sum_{p=1}^P \sum_{\tau_1=0}^{N-1} \cdots \sum_{\tau_p=1}^{N-1} h_p(\tau_1, \dots, \tau_p) \prod_{j=1}^p x(k - \tau_j) \quad (3.12)$$

where  $P$  is the order of the polynomial,  $N$  is the memory depth,  $\tau_p$  are the delays in discrete time and  $h_p$  are the coefficients, often called Volterra kernels [Volterra and Whittaker, 1959]. By increasing  $P$  and  $N$  the accuracy of the model will improve. However, this can add unnecessary computational complexity because the number of parameters will grow exponentially. The Volterra series can also be written as an IIR filter by changing the range of the memory depth. All latter models are simplifications of the Volterra series.

### 3.3.2 Hammerstein Model

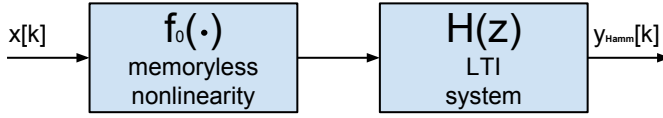
Hammerstein models are composed of a memoryless nonlinear system followed by a linear time-invariant (LTI) system as seen in Fig. 3.2. If the systems are implemented using a memoryless polynomial and a FIR filter respectively, the model can be written as

$$\begin{aligned} y_{Hamm}(k) &= \sum_{n=0}^N \alpha_n f_0(x(k - \tau_n)) \\ &= \sum_{n=0}^N \alpha_n \sum_{p=0}^P \gamma_p x^p(k - \tau_n). \end{aligned} \quad (3.13)$$

The result is two separate sets of linear parameters. The concept of these two-box models is that the two parts are in fact separated. The nonlinear system is not necessarily a memoryless polynomial, it could for example be a Saleh model. Separating the two systems makes an advantage of being able to use all previously known knowledge of memoryless and LTI systems. The Hammerstein model is a common model type.

---

<sup>1</sup>A sum of derivatives to approximate a function around a specific point. More about the Taylor series can be read in Forsling and Neymark [2011].



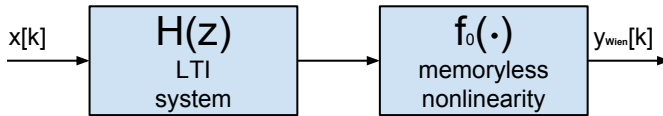
**Figure 3.2:** Block diagram of the Hammerstein model.

### 3.3.3 Wiener Model

The Wiener model is composed of an LTI system followed by a memoryless non-linear system as seen in Fig. 3.3. By switching the order of the systems compared to the Hammerstein model we get the Wiener model as

$$\begin{aligned}
 y_{Wein}(k) &= f_0 \left( \sum_{n=0}^N \alpha_n x(k - \tau_n) \right) \\
 &= \sum_{p=0}^P \gamma_p \left( \sum_{n=0}^N \alpha_n x(k - \tau_n) \right)^p.
 \end{aligned} \tag{3.14}$$

This two-box modeling has a disadvantage compared to Hammerstein since the  $\alpha_n$  coefficients are integrated in the power series making it nonlinear, so the parameter extraction will be more cumbersome. In order to solve it one needs to first estimate an intermediate variable and later solving it in several steps. More about the Wiener model can be read in Luo [2011].



**Figure 3.3:** Block diagram of the Wiener model.

### 3.3.4 Wiener-Hammerstein Model

A three-box modeling can be made by combining a filter both before and after a memoryless nonlinear system. This gives the Wiener-Hammerstein model. All advantages and disadvantages with the two two-box models will comply to the three-box modeling, so the parameter extraction for this model will be even more cumbersome.

The idea of all these box models is to try to capture the memory effects as they occur in the actual physical system. If the origin of the source of memory effects is priorly known, one can take advantage of this to choose an appropriate model type.

### 3.3.5 Nonlinear Moving Average Model

The nonlinear moving average (NMA) model is one of the least complex models that is a simplification of the Volterra series. This model is also called memory polynomial and Fig. 3.4 shows the schematic of the model. It can be written as

$$\begin{aligned}
 y_{NMA}(k) &= \sum_{n=0}^N f_n(x(k - \tau_n)) \\
 &= \sum_{n=0}^N \sum_{p=0}^P \alpha_{pn} x^p(k - \tau_n).
 \end{aligned} \tag{3.15}$$

Comparing this equation to (3.13) which is the example of a Hammerstein model, they can perform the same modeling. As seen the coefficients  $\alpha_{pn}$  are here one set of linear parameters where as for the Hammerstein model it was two sets of parameters. This symbolizes that the Hammerstein is bound to be designed as a model of two systems to be classified as a two-box model.

The modeling capability and simplicity of the NMA model makes it very promising for implementation and use for digital predistortion.

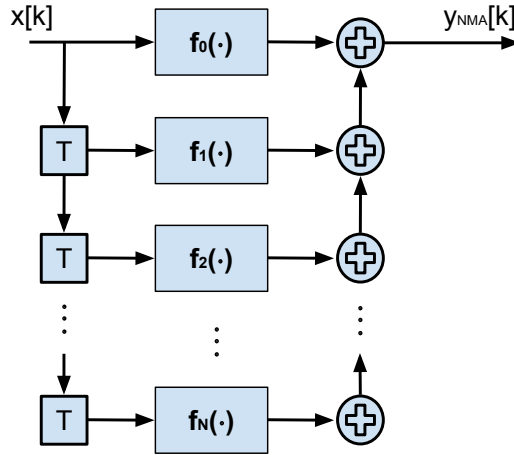


Figure 3.4: Block diagram of the NMA Model.

### 3.3.6 Nonlinear Auto-Regressive Moving Average Model

Extending the NMA model with a feedback path will introduce an auto-regressive part which may relax the order of the moving average part. This will give the nonlinear auto-regressive moving average (NARMA) model. In Fig. 3.5 the schematic

of the NARMA model can be seen. The coefficients  $\alpha_{pi}$  and the value  $N$  in

$$\begin{aligned} y_{NARMA}(k) &= \sum_{i=0}^N f_i(x(k - \tau_i)) - \sum_{j=1}^M g_j(y(k - \tau_j)) \\ &= \sum_{i=0}^N \sum_{p=0}^P \alpha_{pi} x^p(k - \tau_i) - \sum_{j=1}^M \sum_{p=0}^P \beta_{pj} y^p(k - \tau_j) \end{aligned} \quad (3.16)$$

are design parameters for the moving average part. The coefficients  $\beta_{pj}$  and  $M$  are design parameters for the auto-regressive part.  $N$  and  $M$  are memory depths, the  $P$  is the polynomial order. A disadvantage is that with the feedback path it becomes an IIR filter which can result in overall system instability. In order to guarantee the stability of the model a stability test based on the small-gain theorem must hold. More about the NARMA model can be read in Pinal et al. [2007].

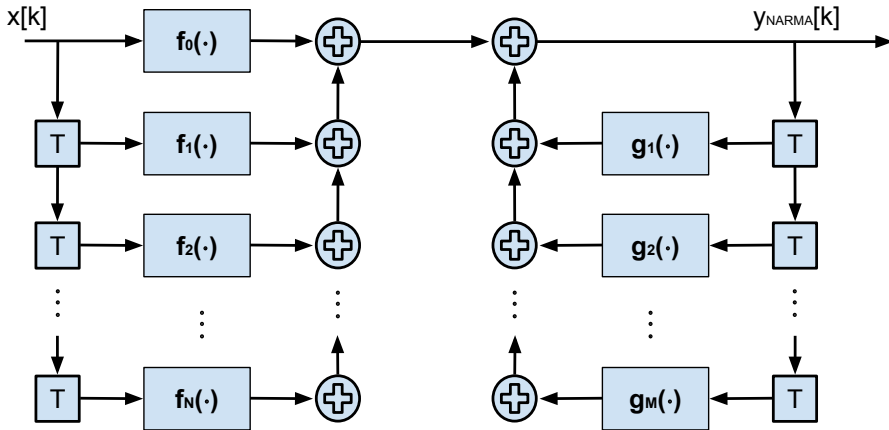


Figure 3.5: Block diagram of the NARMA model.

### 3.3.7 Other Memory Models

There are many other ways to model a nonlinear system apart from the ones previously mentioned. Most of them used for PA modeling is other forms of simplifications of the Volterra series. These includes variations of previously mentioned models such as parallel Hammerstein modeling or augmented nonlinear moving average model [Pinal et al., 2007].

A totally different approach is the artificial neural network modeling. Briefly explained the network tries to emulate the human brain by learning how to behave from previous knowledge. The network is a setup of a large number of basic elements arranged in layers and specific patterns. The connections called links or synapses makes it possible to produce an approximation of any nonlinear function [Mingming et al., 2014].

### 3.4 Parameter Extraction

In all of these models there are parameters to be identified for obtaining a good fit to actual measured data from the true system. The most common way of extracting the parameters is by solving a least squares problem. Other ways could be frequency domain estimation, Lee–Schetzen correlation method and pseudo-inverse technique using the singular-value decomposition. More can be read in Schreurs et al. [2008].

#### 3.4.1 Least Squares Method

For the least squares method it is important that the model can be written with linear parameters as

$$y_{mod}(k) = \varphi^T(k)\theta + e(k) \quad (3.17)$$

where  $\theta$  is the vector containing the linear parameters that are to be determined. The regression vector  $\varphi(k)$  contains the nonlinear terms of the model. If the error  $e(k)$  is considered to be white noise with zero mean the residual can be defined as

$$\epsilon(k, \theta) = y(k) - \hat{y}_{mod}(k; \theta) = y(k) - \varphi^T(k)\theta. \quad (3.18)$$

The idea of the least square method is to form a cost function that is aimed to be minimized by finding the best set of parameters. If the cost function is

$$V_N(\theta) = \frac{1}{N} \sum_{k=1}^N \epsilon(k, \theta)^2 = \frac{1}{N} \sum_{k=1}^N (y(k) - \varphi^T(k)\theta)^2 \quad (3.19)$$

the parameter estimate is defined as

$$\hat{\theta}_N = \arg \min_{\theta} V_N(\theta). \quad (3.20)$$

The fact that  $V_N(\theta)$  is quadratic in  $\theta$  and the parameters are linear makes it possible to have an explicit analytical solution

$$\arg \min_{\theta} V_N(\theta) = \hat{\theta}_N = R_N^{-1} f_N \quad (3.21)$$

where

$$R_N = \frac{1}{N} \sum_{k=1}^N \varphi(k)\varphi^T(k), \quad f_N = \frac{1}{N} \sum_{k=1}^N \varphi(k)y(k). \quad (3.22)$$

As seen in the solution (3.21) it assumes an inverse exists, which means that  $R_N$  has to be positive definite. More about the least square method can be read in Ljung [1999].

#### 3.4.2 Adaptive Least Squares Methods

When it comes to designing hardware it is not desired to perform large matrix multiplications or matrix inversions which is required for the least square method. What is desired and aimed for is minimizing the need of hardware re-



sources and by implementing an adaptive solution this can be achieved. Two adaptive algorithms with different complexity will be examined in this section. A more thorough investigation of these algorithms is presented in Gustafsson et al. [2010].

### Least Mean Square

The least mean square (LMS) algorithm is the simplest of the adaptive algorithms. The goal for the LMS algorithm is that for each iteration minimize the expected value of the cost function

$$V(\theta) = \frac{1}{2}E((y(k) - \varphi^T(k)\theta)^2). \quad (3.23)$$

By differentiating the cost function with respect to  $\theta$ , the negative gradient is obtained as

$$-\frac{d}{d\theta}V(\theta) = E(\varphi(k)(y(k) - \varphi^T(k)\theta)). \quad (3.24)$$

Thus, by moving in the negative gradient with a step-size decided by  $\mu$  one gets a so called steepest descent algorithm as

$$\begin{aligned} \hat{\theta}(k) &= \hat{\theta}(k-1) - \mu \frac{d}{d\theta}V(\theta) \\ &= \hat{\theta}(k-1) + \mu \varphi(k)(y(k) - \varphi^T(k)\hat{\theta}(k-1)). \end{aligned} \quad (3.25)$$

This leads to a simple implementation because for each sample there will just be a couple of multiplications and additions for updating each coefficient. However, the design parameter  $\mu$  has to be small enough for the algorithm to be stable and not to take too long steps. With small steps comes low convergence speed for the final least square solution.

### Recursive Least Square

Both the recursive least square (RLS) and the LMS algorithm are variations of a Kalman filter<sup>2</sup> on the form

$$\hat{\theta}(k) = \hat{\theta}(k-1) + K(k)\varphi(k)(y(k) - \varphi^T(k)\hat{\theta}(k-1)) \quad (3.26)$$

where the Kalman gain  $K(k)$  divides them apart. The LMS algorithm has the Kalman gain as a constant parameter  $\mu$  whereas the RLS algorithm has a more complex approach. By computing  $R_N$  and  $f_N$  from (3.22) recursively and further introducing  $R^{-1}(k) = P(k)$  we can with the matrix inversion lemma<sup>3</sup> write the Kalman gain as

$$K(k) = \frac{P(k-1)\varphi(k)}{\lambda + \varphi^T(k)P(k-1)\varphi(k)}, \quad (3.27)$$

<sup>2</sup>A recursive algorithm named after Rudolf E. Kalman. More can be read in Gustafsson et al. [2010].

<sup>3</sup> $(A + BCD)^{-1} = A^{-1} - A^{-1}B(DA^{-1}B + C^{-1})^{-1}DA^{-1}$  [Gustafsson et al., 2010].

where

$$P(k) = \frac{1}{\lambda} \left( P(k-1) - \frac{P(k-1)\varphi(k)\varphi^T(k)P(k-1)}{\lambda + \varphi^T(k)P(k-1)\varphi(k)} \right). \quad (3.28)$$

The so called forgetting factor  $0 < \lambda \leq 1$  is a design parameter that inflects on how fast old measurements in the cost function are forgotten as time evolves. What is gained from the added computation complexity is a faster convergence speed as well as a guaranteed stability. However, if convergence speed is not highly prioritized there is extra computation for each iteration compared to the LMS algorithm. The need to add a unit for division is also imposed.

### 3.5 Conclusions

The main concern when selecting a model is the trade-off between complexity and accuracy. It will latter be shown that the used PA suffer from memory effects, the memoryless models give poor results and are therefore excluded. The more complex NMA and NARMA model is designed to deal with these memory effects and better results are achieved. As for the different box models they do not add anything of additional value compared to the NMA and NARMA model, the parameter extraction is also more cumbersome. However, for some applications the box models could have an advantage of needing less number of parameters. This is true when a good partitioning can be done between the boxes. More about the model selection can be found in Section 6.1.

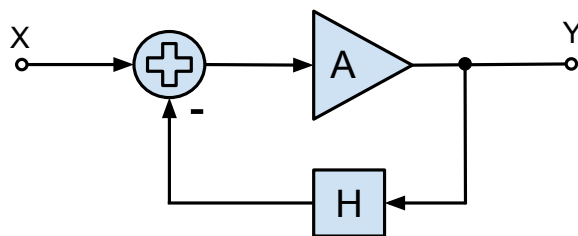
# 4

## Linearization Techniques

As the name suggests, a linearization technique is a technique that makes a (non-linear) system behave as if it was linear. The focus will be linearization of PAs, since that is the topic of this thesis. However, most of the techniques in this chapter can be used for arbitrary nonlinear systems.

### 4.1 Feedback

Feedback is a well known technique that has been used for a long time in control theory. The concept can be used to improve the linearity of the PA. The general feedback closed loop system is shown in Fig. 4.1.



*Figure 4.1: Feedback block diagram.*

From the figure the output can be determined as

$$Y = \frac{A}{1 + AH} X \quad (4.1)$$

where  $X$  is the input,  $A$  is the gain of the PA and  $H$  is the feedback loop transfer function. Assuming that  $A$  is large and  $H$  is not too small gives

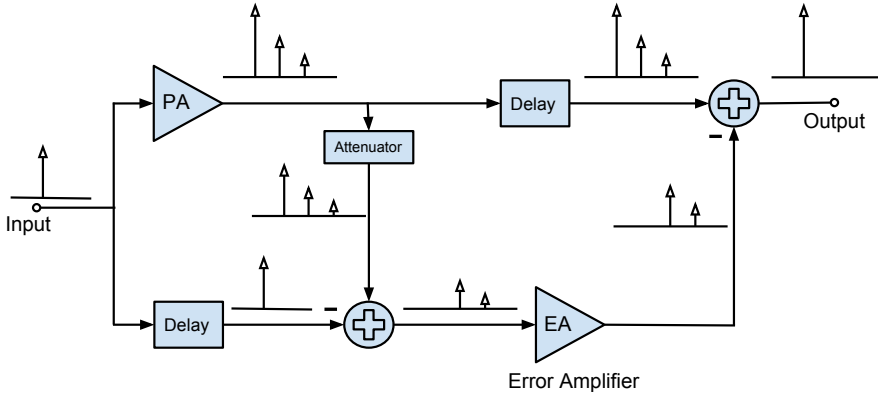
$$Y \approx \frac{A}{AH} X = \frac{1}{H} X. \quad (4.2)$$

This means that the system is insensible to variation in gain. However, the price for this is that the total gain of the system is reduced.

One of the major problems with the feedback technique is stability. As seen in (4.1) the system will become unstable e.g. when  $AH = -1$ . To avoid this problem it is very important to have enough phase and gain margin in the feedback path. There are many different variations of the feedback technique used as linearizers, such as *Cartesian feedback*, *polar loop feedback* and *envelope feedback*. More about the different feedback techniques can be read in Pinal et al. [2007].

## 4.2 Feedforward

In the feedforward technique the correction is done at the output compared to the feedback technique where the correction is done at the input. The benefit of this is that the system does not suffer from instability.



**Figure 4.2:** Feedforward block diagram.

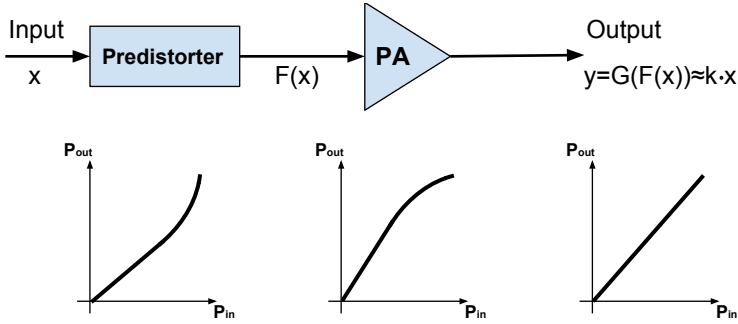
The basic idea is shown in Fig. 4.2 with a one-tone input. As seen, the input is first divided into two paths where one of them is fed to the PA and the other one is delayed. The output of the PA, which will include distortions, is then attenuated and subtracted to the delayed input. Ideally this will leave only the distortions in the lower path. This is then amplified to the same level as the upper path by another amplifier, called the error amplifier (EA). Finally the two paths are subtracted from each other, ideally leaving only the signal at the output.

However, there are some drawbacks with this technique. First of all an additional

amplifier is needed which has to be linear enough to not add to much distortion on its own. Also the delays in the upper and lower paths have to be calibrated to match the delays in the PA and EA respectively to not degrade the performance. [Pinal et al., 2007] [Cripps, 2006].

## 4.3 Predistortion

The idea behind predistortion is quite simple. Instead of feeding the input signal directly to the PA it is first fed through a block called the predistorter. The predistorter is a nonlinear system that counteracts with the PA's nonlinearities. Ideally the predistortion block is the inverse of the PA which will make the cascaded output of the two blocks linear. This is shown in Fig. 4.3 with  $x$  as the input and with  $F(\cdot)$  and  $G(\cdot)$  as the predistorter and PA function respectively.



*Figure 4.3: The predistortion concept.*

However, finding the inverse of a nonlinear system is not a trivial task. First of all the system has to be bijective, i.e. there must be a one-to-one mapping between the input and the output. To find the inverse analytically is usually very difficult and sometimes not even possible [Söderkvist and Ahnell, 1994]. If the analytically inverse is not possible to find, it can be estimated. Different techniques to find this estimation will be discussed in the upcoming sections. Predistortion can be done both in the analog and the digital domain. Though this thesis mainly focus on digital predistortion, the analog technique will also be mentioned.

### 4.3.1 Analog Predistortion

Most analog solutions have a longer history than the digital solutions for natural reasons. But knowledge from analog predistortion can be taken into account and used for newer reformed digitized solutions.

### Cubic Predistortion

One common method for analog predistortion is called cubic predistortion. In this technique the input signal is split into two paths. One of the paths consist of a cubic nonlinearity which will perform the nonlinear predistortion. The other path is simply the input, delayed so that it matches the first path. The distorted path is then subtracted from the delayed input before fed to the PA. The nonlinearity is usually created by using components with a nonlinear behavior such as the diode.

As the name suggests, this technique is used to suppress the third order IM products. Sometimes it is used together with a feedforward structure to improve linearity further [Pinal et al., 2007].

### Harmonic Feedback Predistoriton

Another analog solution is harmonic feedback, also called harmonic injection. One can take advantage of the output distortions by feeding the output signal back and add them to the PA input for linearization purposes. First the output signal goes through a selective filter to extract a harmonic to be calibrated. The calibration is done by letting the feedback path consist of a variable phase shifter and a variable gain amplifier. With optimum settings this may reduce specific distortions, usually the IM3. More on this can be read in Moazzam and Aitchison [1996]. One disadvantage is the need of calibration and that there is no clear way of finding the optimum settings.

### 4.3.2 Digital Predistortion

Digital predistortion is implemented on a digital system. It can be designed either using a look-up table (LUT) or calculated using a model. The LUT in this case is simply a memory that maps the input to a predistorted output that will be fed to the PA. With a model it is basically the same but the predistorted output has to be calculated for every input sample. The two implementations have different advantages and disadvantages. As the resolution of the DPD gets higher the LUT size gets bigger and it will consume more area. However, the model implementation will have to do calculations continuously and will therefore consume more power.

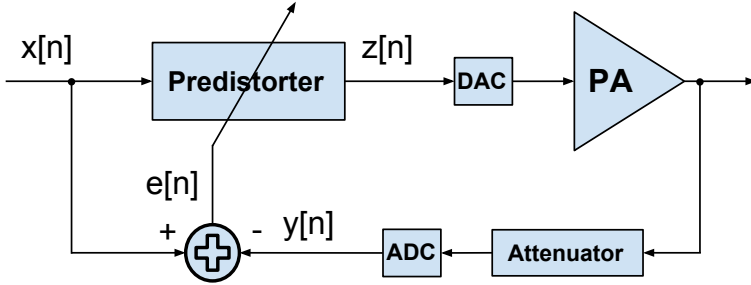
To get the right LUT values and coefficients for the model an examination of the PA's characteristics has to be done in advance. An estimation of these parameters can be found using the LS method (Section 3.4.1).

Most of the techniques used today use some kind of feedback making the predistortion adaptive. This makes a more robust solution since it can deal with changes in the PA's characteristics. These changes may come from temperature variations, power supply voltage variations, aging etc. With the feedback the LUT values and model coefficients can be found using an adaptive algorithm such as the RLS or the LMS (Section 3.4.2).

There are two different architectures for adaptive predistortion (APD); direct learning architecture and indirect learning architecture.

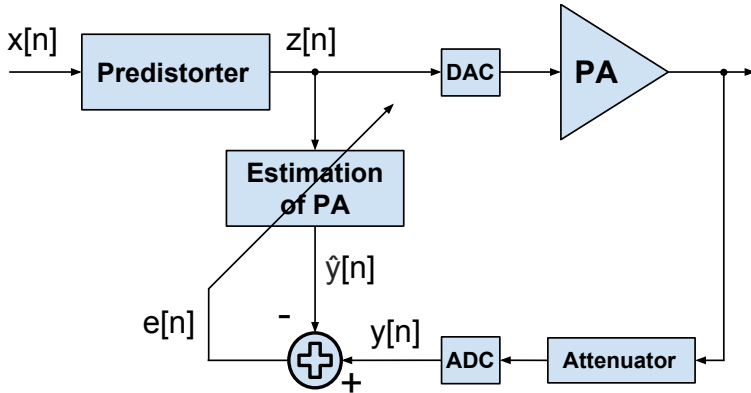
### Direct Learning Architecture

The basic structure of the direct learning architecture (DLA) is shown in Fig. 4.4.



**Figure 4.4:** Direct learning architecture, type 1.

Here  $x[n]$  is the input,  $z[n]$  is the output of the predistorter and  $y[n]$  is the normalized output of the PA. The error,  $e[n]$ , is the difference between the input and the normalized PA output. This value will update the parameters in the predistorter and go towards zero as the PA output gets more linear. However, the problem with this structure is that there is no direct relationship between the error and the predistorter model parameters. To work around this problem, the structure in Fig. 4.5 can be used.

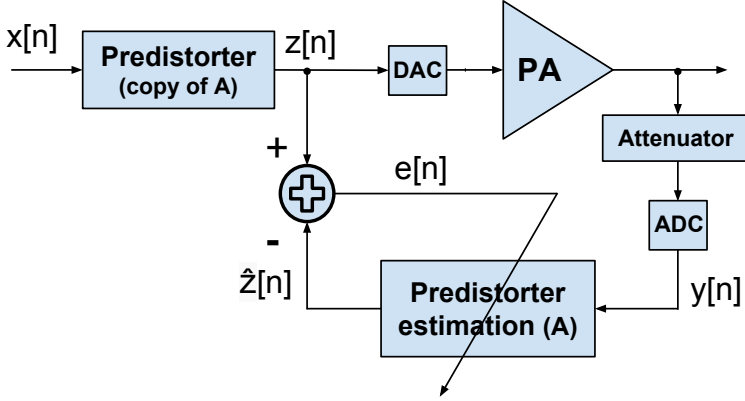


**Figure 4.5:** Direct learning architecture, type 2.

Here a model of the PA is introduced which will be updated by the error signal,  $e[n]$ . The error,  $e[n]$ , is the difference between the real PA output and the estimated PA output. As the estimation of the PA gets better the error will go towards zero. The ideal solution is when  $\hat{y}[n] = y[n]$ . The problem with the second DLA structure is that it only gives an estimation of the PA. To find the predistorter parameters an inverse of the PA estimation has to be calculated [Luo, 2011].

### Indirect Learning Architecture

The indirect learning architecture (ILA) has the same idea as the DLA but here the predistorter is estimated. Therefore the structure learns about the PA indirectly (hence the name). Figure 4.6 shows the architecture.



*Figure 4.6: Indirect learning architecture.*

The normalized output from the PA,  $y[n]$ , is fed to a predistortion block with  $\hat{z}[n]$  as output. The input signal,  $x[n]$ , is fed to another predistorter block with output  $z[n]$ . The error signal,  $e[n]$ , is defined as  $e[n] = z[n] - \hat{z}[n]$ . The predistorter parameters can then be retrieved by minimizing  $e[n]$ . Worth noting is that the predistorter in the upper path then can be a copy of the estimated predistorter [Luo, 2011].

## 4.4 Conclusions

For this thesis the digital predistorter linearizer is used. The main reason for this is to avoid the stability issues that comes with a feedback system and also to avoid an extra PA which is needed in the feedforward case.

Two different variations of the digital predistortion technique is implemented to compare them with each other. Both are made adaptive to be able to cope with changes in the PA's characteristics. The first implementation uses a functional model as predistorter. To extract the parameters for the model the ILA is chosen. The reason for this is that ILA extracts the predistortion parameters in a direct way. The second approach is model-free and instead it uses a LUT as the predistorter. To find the parameters a digitized version of the harmonic feedback is used. These two approaches will be discussed more closely in Chapter 6 and 7.



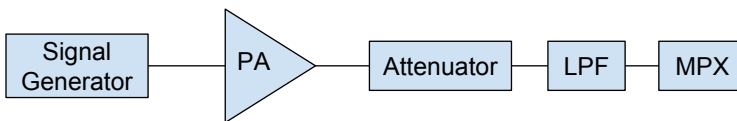
# 5

## Measurements

Measurements had to be done both to be able to create nonlinear models of the PA and also later to measure the performance of the DPD. This chapter presents the measurement setup as well as the different components and tools used. It also shows results from the measurements.

### 5.1 Measurement Setup

To measure the PA characteristics a basic setup was used, shown in Fig. 5.1.

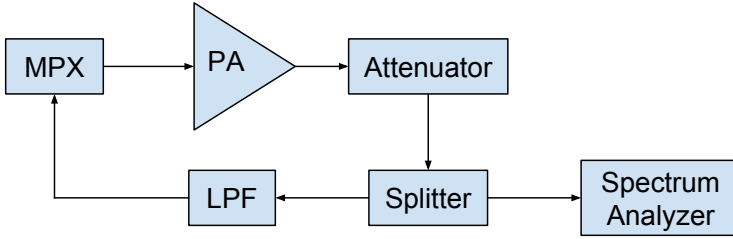


*Figure 5.1: Basic measurement setup.*

The signal generator was a Rhode & Schwarz vector signal generator SMBV100A. The PA used (and the one that was linearized) was an Empower BBM2E3KKO. This is a 100 watt Class AB PA working in the range of 20-520 MHz with a 50 dB gain and  $P_{out,1dB} = 60$  watt. The data sheet specified that the third harmonic would be -15 dBc at 100 watt output power. However, as the measurement result will show, the performance was actually worse than that. After the PA an attenuator was used to lower the signal power. The attenuation was 50 dB (same as the gain of the PA) making the output the same level as the input. To collect the data an MPX card was used. This is a card made by FOI that can be used to transmit and receive data. The ADC used in the receiver part was clocked at

409.6 MHz. This limited the sampled input signal to ideally only contain frequency components below 204.8 MHz (Nyquist frequency). To reduce aliasing a lowpass filter was used before the MPX card. However, an analog filter is not ideal, meaning that the transition from passband to stopband is performed over a certain frequency range. Therefore, a cut-off frequency of 200 MHz was used for the lowpass filter.

When the DPD was implemented a new setup was used to measure the performance. This is shown in Fig. 5.2.



**Figure 5.2:** Setup to measure the DPD performance.

Two components were added to this setup. First a spectrum analyzer (Rhode & Schwarz FSH3) to examine the data in real-time. A splitter was also added to be able to use the spectrum analyzer and run the system at the same time. The splitter attenuates the signal an additional 3 dB.

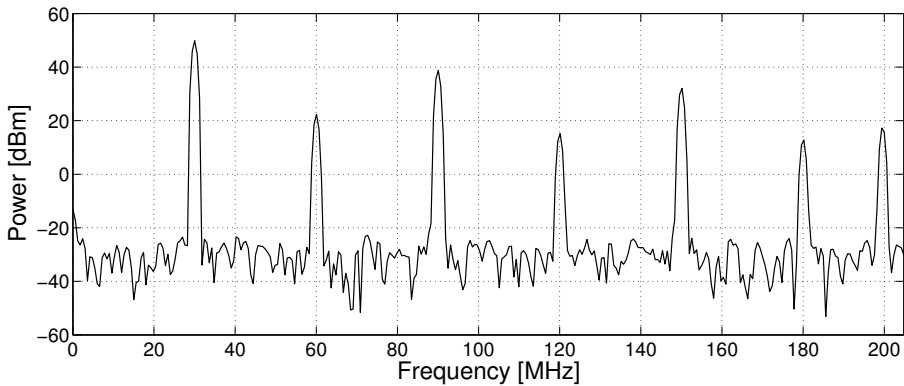
## 5.2 Measurement Results

To describe an absolute signal level the quantity dBm is often used. It is the power level of the signal referenced to 1 mW, i.e.

$$P_{sig}|_{dBm} = 10 \log_{10} \left( \frac{P_{sig}}{1mW} \right). \quad (5.1)$$

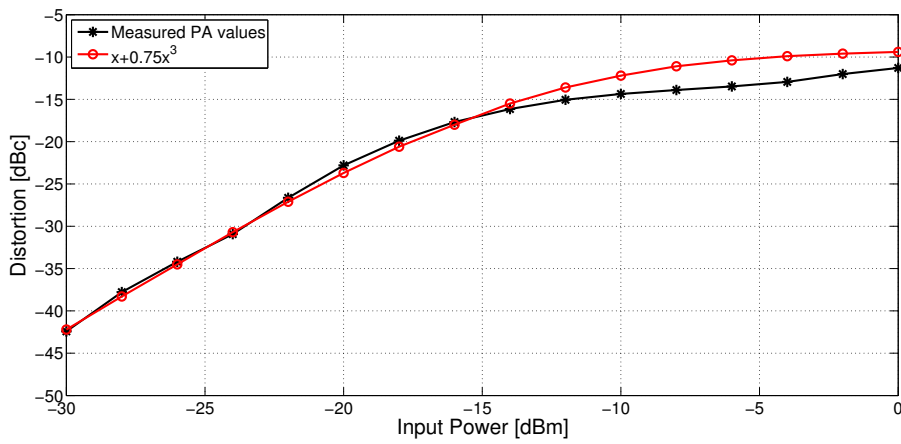
To examine the nonlinear behavior of the PA, a 30 MHz sine wave was applied to the input. The power of the input signal was 0 dBm. The discrete Fourier transform (DFT) of the output signal is shown in Fig. 5.3. There is a lack of samples because the data collection is done in the FPGA design with the Xilinx chipscope modules. Later in simulations the number of samples will be set to 8192. Throughout this thesis the Blackman window is used as the selected window function. As seen in the DFT plot the PA adds distortion. The largest harmonic is as expected the third harmonic, which is around -11 dBc. Also the fifth harmonic is quite large. Worth noting is that the last tone is occurring due to aliasing. The seventh harmonic should be at  $7 \cdot 30 \text{ MHz} = 210 \text{ MHz}$  but since the Nyquist Frequency is lower (204.8 MHz) the tone is instead shown at  $2 \cdot 204.8 - 210 = 199.6 \text{ MHz}$ . Ideally the lowpass filter should have taken care of

this but since it is not ideal some power still got through.



**Figure 5.3:** DFT of the output from the real PA when applied with a 30 MHz sine input.

Figure 5.4 shows how the distortion of the system change for different input power, measured in dBc. A 30 MHz sine wave was applied to the input. Two sets of data is shown, one for the PA and one for a simulated PA model. The measurements of the PA showed that for each input power selected the total distortion was set by the dBc value related to the third order harmonic, since it is the largest source of distortion. As seen in the figure, and as expected, the dBc decreases when the input power gets lower. This means that the signal gets more linear for lower inputs. It is possible to get a value of -40 dBc without any kind of predistortion but then the input has to be lowered with around 29 dB. One can also note that the dBc values for the PA gets saturated for higher inputs. Comparing the data for the PA with the data of the PA model they have clearly similar characteristics. This supports the fact that the third order harmonic is the biggest source of distortion since the only nonlinear part of the PA model is a third order term. However, this is by no means a model that fully captures the nonlinear behavior of the PA, since it is such a complex system.



**Figure 5.4:** The distortion of the PA as a function of input power. Data from the PA as well as a simulation of a simple model, with a third order term, is shown. It is clear that the two datasets has similar characteristics thus supporting the fact that the third order harmonic is the largest source of distortion.

# 6

---

## Simulations

This chapter will investigate two different variations of the digital predistortion technique. Simulations carried out in MATLAB will be presented to verify the functionality. Different design parameters will be simulated to see how they affect the overall performance on the total system.

### Normalized Mean Square Error

The normalized mean square error (NMSE) can be used to calculate the difference between two signals. NMSE compares a estimated signal  $\hat{y}$  with a true reference signal  $y$ . It is calculated using

$$\text{NMSE} = 10 \log_{10} \left( \frac{\sum_{n=0}^{N-1} |\hat{y}(n) - y(n)|^2}{\sum_{n=0}^{N-1} |y(n)|^2} \right) \quad (6.1)$$

where  $N$  is the number of samples. Since NMSE calculates the error it should be as low as possible. In this thesis the NMSE and dBc measurements are often used to measure the performance of linearization as well as PA modeling. A difference between the measurements is that NMSE is calculated in the time domain whereas the dBc measurement is used in the frequency domain. This give a somewhat complementary measurement.

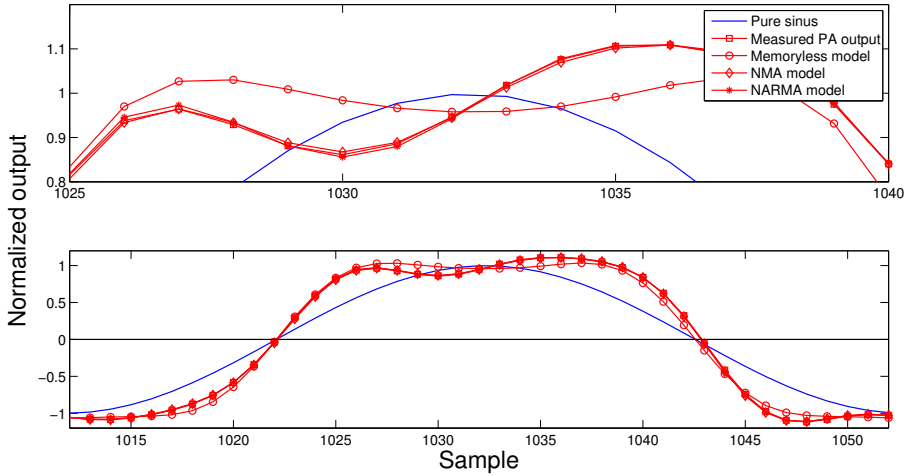
### 6.1 Power Amplifier Modeling

The main purpose of modeling the PA is to be able to use it in simulations to test the different linearization techniques. Table 6.1 shows the results when modeling the PA with a memoryless polynomial model ( $P = 5$ ), an NMA model ( $P = 5, N =$

1) and a NARMA model ( $P = 5, N = 1, M = 1$ ). The models have been extracted from measured data on the real PA. As seen, the NMA and NARMA is much better than the memoryless model which means that the PA clearly suffers from memory effects. This is also shown in Fig. 6.1 where the curve fitting from the different models is visible. Even though it is hard to tell the difference between the NMA and the NARMA model, one can easily see that the memoryless model is the worse of the three since it has to be symmetric around the extreme point.

	Memoryless	NMA	NARMA
NMSE	-22.35 dB	-48.64 dB	-52.31 dB

**Table 6.1:** Model investigation of memoryless polynomial model, NMA model and NARMA model compared with NMSE.

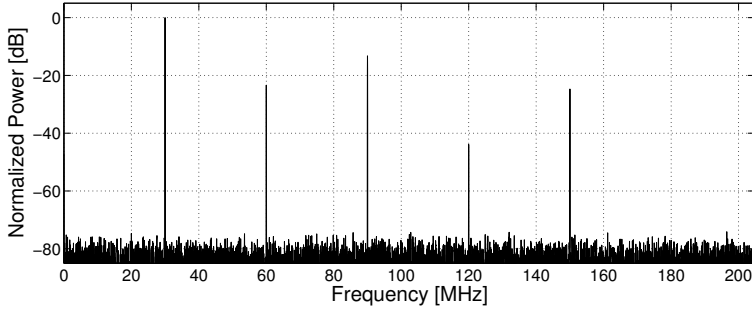


**Figure 6.1:** Curve fitting from the model investigation of memoryless polynomial model, NMA model and NARMA model. The upper plot is a zoomed in version of the lower plot.

By just comparing the two memory models it seems like NARMA would be the best choice to model the PA because it has slightly lower NMSE value. However, since the NARMA model suffer from stability issues the NMA model is chosen instead. The coefficients extracted for the NMA are

$$\begin{aligned} a_1 &= 2.4757 & a_2 &= -0.2804 & a_3 &= -5.3079 & a_4 &= 0.0364 & a_5 &= 4.8113 \\ a_6 &= 0.3714 & a_7 &= 0.3148 & a_8 &= -2.4963 & a_9 &= -0.1351 & a_{10} &= 3.7064 \end{aligned} \quad (6.2)$$

These coefficients will be used throughout this chapter for the PA model. However, the LUT simulations will use the memoryless model as well. To make the conditions more realistic a -80 dBm white noise signal is added to the PA output. The DFT of the default output from the NMA PA model is shown in Fig. 6.2.



**Figure 6.2:** DFT of the NMA PA model output with a 30 MHz one-tone signal as input.

## 6.2 Motivation and Decisions

The goal of this thesis is to implement a linearization system that uses digital predistortion (Section 4.3.2) on an MPX card. This technique is chosen because the FPGA can be further utilized when generating the input signals for the PA. There is no need for extra hardware like PAs as in the feedforward linearization technique (Section 4.2). The digital predistortion can be implemented in many different ways, but the goal is always the same, to find the input that will generate the most linear output. The synthesis of this input signal can be done through a functional model or without a model using a LUT. A decision was made to implement one of each approach, so that comparisons between the two approaches can then be performed.

The extraction of the model parameters and the LUT content is decided to be carried out online on the FPGA. An offline extraction could be possible with an accurate nonlinear model of the PA. However, the model would never be as good as using the true PA motivating the choice of online extraction. This decision means an introduction of a feedback path. The actual extraction will differ between the two implementations.

### 6.2.1 Model Implementation

As mentioned before, choosing a model type and its design is a trade-off between complexity and accuracy. What is seen from previous work such as Luo [2011], is that the NMA model is popular and has good trade-off between these properties. It has a straightforward parameter extraction without rewriting the coefficients to intermediate variables as for the box models (Section 3.3.2). The NMA also has a memory modeling property, and choosing a more complex model like NARMA will introduce the risk of instability. The result in Section 6.1 further motivates the choice of the NMA model.

To extract the parameters of the predistorter unit, the ILA (Section 4.3.2) is chosen as the best alternative. The ILA is simpler than the DLA because the inverse parameters can be extracted in a direct way.

### 6.2.2 Model-free Implementation

Creating and designing a nonlinear model is a complex procedure and a model has its limitations. Based on harmonic feedback predistortion a model free implementation has been designed. The implementation makes use of a LUT to synthesize the predistortion signal. The design is specialized and limited to work for the used jamming technique. For each frequency used when jamming there has to be a corresponding LUT that contains the distortion parameters. The extraction of a LUT and its parameters is conducted with a slow adaption process, where the current error of each parameter is adjusted for. More about the model-free implementation and its details will be presented in Section 6.4.1. This design will most probably use more memory but less functional hardware. If this is true it can have an advantage of lower power consumption due to less switching activity<sup>1</sup>.

### 6.2.3 Input Signal

There are several techniques and ideas for how to generate signals for a jamming system. But for this thesis the input signal for implementation is limited to single frequencies, i.e. one-tone signals. Together with frequency hopping over wide bandwidth this makes up a jamming system. However, for showing generality of the digital predistortion technique, two-tone signals will also be used in the simulation stage.

## 6.3 NMA Simulation

The model order for the NMA predistorter was chosen using Figure 5.3. This measurement was done with the lowest desired input frequency, 30 MHz, so the maximum numbers of harmonics were visible. But even though the sixth and seventh harmonics existed (seventh only because of aliasing) their power was so low that they could be ignored. The fifth harmonic had a power of more than the desired -40 dBc. Therefore a model order of five looked like the best choice. However, because of the low power in the fourth harmonic, the fourth-order term was not really necessary in the model. The choice was to use a model order of five but excluding the fourth-order term to save hardware. This gave the final NMA model as

$$f_{NMA}(x) = x(k) + x^2(k) + x^3(k) + \dots + x^3(k - \tau) + x^5(k - \tau) \quad (6.3)$$

where  $x$  is the input to the model and  $\tau$  is the memory length.

As mentioned in Section 6.2.1 the ILA was chosen for the parameter extraction. To make this work the NMA model was first rewritten as a linear regression model according to (3.17). The normalized output of the PA,  $Y_t$ , is the input

---

<sup>1</sup>  $P_{dyn} = C_L V_{DD}^2 P_{0 \rightarrow 1} f$  is the dynamic power consumption in a digital system where the activity factor,  $P_{0 \rightarrow 1}$ , is a number between zero and one that represent the activity of switches in the logic gates. More about power consumption in Rabaey et al. [2003]



to the system which gives the following:

$$\begin{aligned}\varphi(k) &= (Yt(k), Yt^2(k), Yt^3(k), Yt^5(k), Yt(k-1), \dots, Yt^5(k-\tau))^T \\ \theta &= (a_1, a_2, \dots, a_N)^T\end{aligned}\quad (6.4)$$

where  $\tau$  defined the memory depth and  $N$  the number of coefficients. The residual is then defined as

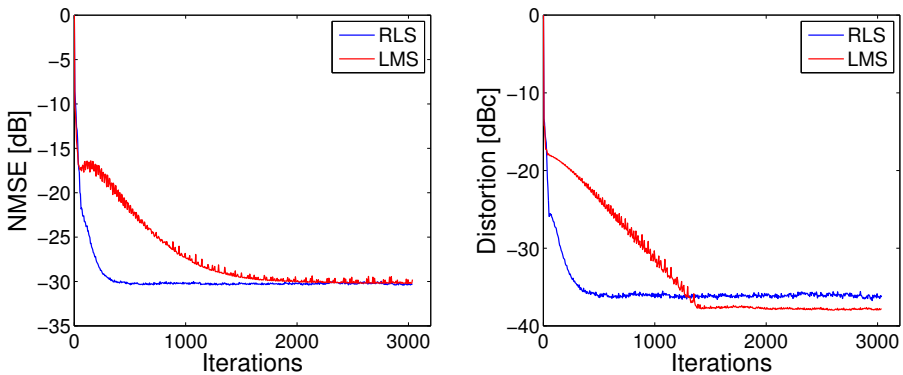
$$\epsilon(k) = Zt(k) - \varphi^T(k)\theta \quad (6.5)$$

where  $Zt(k)$  is the output of the DPD system.

The adaptive algorithms used and compared in the simulations are the LMS and the RLS.

### 6.3.1 Results

At first, simulations were made to confirm that the concept worked and also to compare the LMS and RLS algorithm. The model of the PA was chosen according to (6.2). The DPD used a model according to (6.3) with  $\tau = 2$ . Figure 6.3 shows the results for the two adaptive algorithms, both in terms of NMSE and dBc.

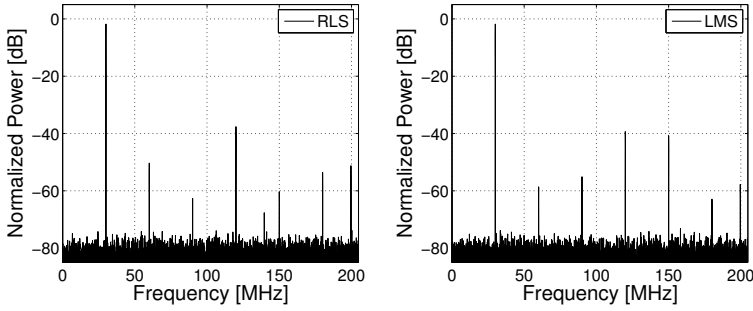


**Figure 6.3:** Comparison between the LMS and the RLS algorithm.

As can be seen in the figure the performance has been greatly improved for both the cases. The RLS converge faster than the LMS, but the price for this is that the RLS has much higher computational complexity. In the NMSE measure they converge to the same value which also is expected since the algorithms are calculated in the time domain. However, in the frequency domain the LMS actually shows better results after around 1200 iterations. The RLS converge at  $\sim 36$  dBc whereas the LMS goes down an additional 2 dB.

The DFT of the linearized outputs for both RLS and LMS are shown in Fig. 6.4.

Compared to the default output of the PA (Fig. 6.2) the fundamental tone is almost 2 dB lower for both the algorithms. Even though the DPD and the PA model only consists of orders up to five, both the sixth and seventh (due to aliasing) harmonic are visible. That is because the DPD and the PA models are cascaded. For



**Figure 6.4:** Linearized result for RLS (left) and LMS (right) when simulating a 30 MHz one-tone signal.

example, when the  $x^3$  term from the DPD output goes through the PA model the term  $(x^3)^3$ , among many others, will be created. This term will add energy to every odd order harmonic up to the ninth. If one look closely, the aliased ninth harmonic can be seen in the RLS solution between the fourth and fifth harmonic.

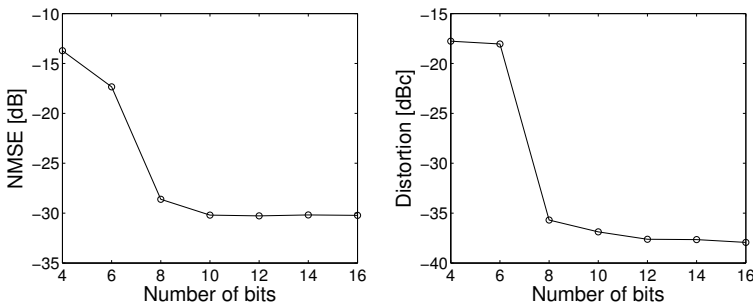
### Bit Length

Since the algorithm was to be implemented on an FPGA with fixed point precision, simulations were carried out with different bit lengths for the coefficients. This meant that the coefficients had to be quantized in every iteration. The following was used in MATLAB for the quantization:

$$a_{quant} = \text{round}(a \cdot 2^q) \cdot 2^{-q} \quad (6.6)$$

where  $a$  was the coefficient to be quantized and  $q$  represented the number of bits used in the quantization. Worth noting is that the command round was used in MATLAB. To get the same behavior in the implementation, proper rounding had to be performed.

The results from the bit length simulations are shown in Fig. 6.5. In all the cases the LMS algorithm was used. The DPD and PA model were the same as the previous simulation.

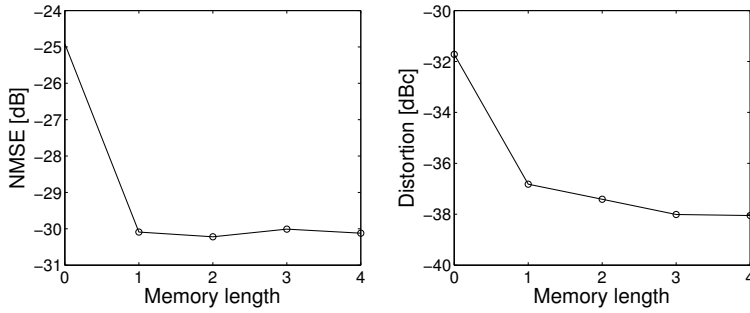


**Figure 6.5:** Performance of the system with quantized coefficients.

As seen, the largest performance change comes when going from 6 to 8 bits. After 10 bits the NMSE stays basically the same. The dBc, however, continues to decrease slowly until 38 dBc is reached at 16 bits.

### Memory Length

Deciding the memory length  $\tau$  of the DPD is crucial since every new memory added will introduce more coefficients and thus more hardware will be needed in the FPGA. Therefore simulations were carried out with  $\tau = 0, 1, 2, 3, 4$ . Again the LMS algorithm and the same PA coefficients were used. The results are shown in Fig. 6.6.



**Figure 6.6:** Performance of the system for different memory lengths,  $\tau$ .

As expected the model without any memory terms gives the worst performance. When adding memory terms the NMSE converge to around -30 dB. Adding up to 3 memories will give better dBc. However, at 4 memories the dBc saturates.

### Variations of $\mu$

An important issue to consider when using the LMS algorithm is to not use a too large step size  $\mu$  value because of stability reasons. Table 6.2 shows the results for five different  $\mu$  values.

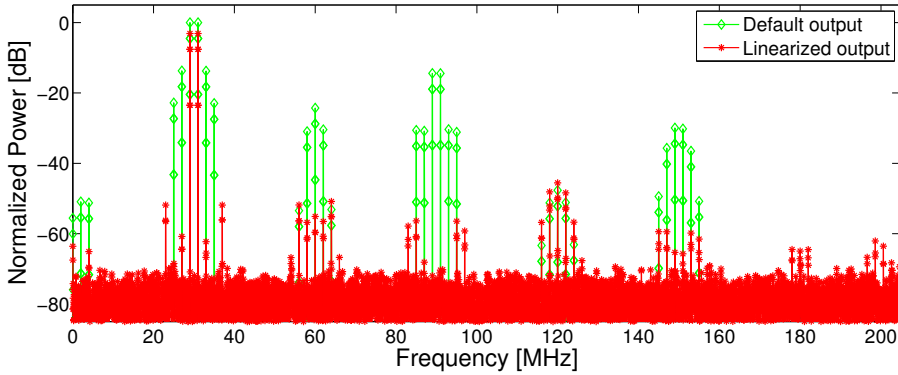
$\mu$	NMSE	dBc
1	-22.87	-31.74
0.5	-29.96	-37.68
0.25	-30.2	-37.66
0.125	-30.05	-37.34
0.0625	-29.91	-37.42

**Table 6.2:** Performance of the system for different step sizes  $\mu$ .

Worst performance is given by  $\mu=1$  which is an indication that the value is too large. For all the other step sizes the results are basically the same. The small difference in NMSE and dBc in these cases originates from the added white noise at the PA model output.

## Two-tone Input

A two-tone signal was used as input to see how the predistorter reacted to it. The frequencies used were  $f_1 = 29$  MHz and  $f_2 = 31$  MHz. The default output from the PA as well as the linearized output are shown in Fig. 6.7. As seen, the predistorter works fine with more than one input frequency. The dBc is almost as good as for the one-tone tests, close to -40 dBc.



**Figure 6.7:** DFT of the NMA linearization result when simulating a two-tone on the memory PA model using  $f_c = 30$  MHz and  $\Delta f = 1$  MHz.

## 6.4 LUT Simulation

The LUT method that is investigated is free from a nonlinear model. The method is also specialized and limited to work for the used jamming technique i.e a one-tone signal. A predistortion signal compound of the correct frequencies, amplitudes and phase will give a desired linearized output. But finding this signal is a troublesome task. The frequencies for the predistortion signals are known, these will be the harmonics within the given bandwidth. Finding the amplitude and phase for each harmonic could be done by searching for amplitudes and phases that will perform a better output. However, there is no good decision rule for making this search.

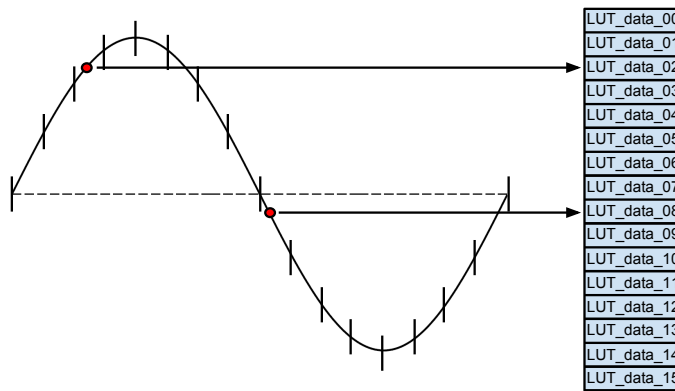
The investigated method will be a digitized version of the analog technique of harmonic feedback predistortion, see Section 4.3.1.

### 6.4.1 Method

The main idea of this method is to take advantage of the harmonics produced by the nonlinear system by feeding them back. Subtracting the signal that is fed back with the sent pure fundamental tone will give a signal that only contains the harmonic distortions in anti-phase. This signal is then used to produce a predistortion signal.

As mentioned before the definition of a nonlinear system is that the superposition

do not hold. However, in this method an assumption is made that a more linear result is expected when a scaled anti-phase signal is feed through a nonlinear memoryless PA model. This is because it has only amplitude distortion and no phase distortion. Further on slowly iterating this process with a scaled anti-phase signal will give the final predistortion signal. A solution made in the time domain has showed that this works reasonable even when phase distortion is present, thus giving a more linear output. For one given frequency there will be one specific adapted LUT, slightly changing the frequency to nearby frequencies will decrease the performance. This means that to perform jamming over various frequencies at a fast pace it would be preferred to have a fast adaption algorithm. A complement to this is to perform an initiation that stores a bank of LUTs where each LUT is adapted for one small range of frequencies.



**Figure 6.8:** Example of the mapping between LUT and sinusoid signal with  $N = 16$ . A sample within one segment of the signal belongs to a specific cell in the LUT. In each cell a distorter parameter is stored as well as a boolean for knowing if the parameter has been updated.

In Fig. 6.8 a LUT is constructed with  $N = 16$  cells and a pure sinusoid is divided in  $N$  equal segments mapping each memory cell to one segment. Each cell can then be addressed by the amplitude and the sign of the derivative. The sinusoid is of a chosen frequency and in the memory cells distortion parameters can be saved. The goal is that the whole LUT with its distortion parameters makes up the feedback anti-phase predistortion signal. This is constructed sample by sample, subtracting a pure sinusoid sample with an output sample and later scaling it down to be stored in its memory cell. Updating all the memory cells will then give a LUT containing the scaled anti-phase signal. One iteration has now been made and a predistortion signal different from the pure sinusoid can be sent through the PA.

Having a low scale-factor and many iterations will make a slow adaption of the final predistortion signal. At each iteration the distortion parameters are accumulated. This method can be viewed as a correction for each distortion parameter by using the current error in each iteration, where the current error is the difference

between the samples of the pure sinusoid and the PA output. This means that all the stored distortion parameters are handled in a somewhat independent way. If a memory PA model is simulated a worse result should be expected than for a memoryless PA model. This is because the memory PA model adds phase distortion.

### Algorithm

To summarize the method to extract a LUT for one specific frequency a pseudo code algorithm is presented:

```

1.  set_scale_factor( $\mu$ );
2.  set_LUT_size( $N$ );
3.  create_LUT_data(LUT_size);
4.  create_LUT_data_tmp(LUT_size);
5.
6.  while true
7.      % Predistorter part
8.       $X_{in\_derivative} = \text{sign}(X_{in}[n] - X_{in}[n - 1])$ ;
9.      set_cellpointer( $X_{in\_derivative}, X_{in}[n]$ );
10.     get_LUT_value(LUT_data, cellpointer);
11.      $Z_{out}[n] = X_{in}[n] + LUT\_value$ ;
12.     % Power amplifier
13.     get_Y_out( $Z_{out}$ , PA_coefficients);
14.     % Adaptive part
15.      $error = X_{in}[n] - Y_{out}[n]$ ;
16.      $scaled\_error = scale\_factor \cdot error$ ;
17.     is_parameter_updated(LUT_data_tmp, cellpointer);
18.     if parameter_updated == false
19.          $LUT\_value\_new = LUT\_value + scaled\_error$ ;
20.         save_LUT_value(LUT_value_new, LUT_data_tmp, cellpointer);
21.         save_parameter_updated(LUT_data_tmp, cellpointer);
22.     end if
23.     is_all_parameters_updated(LUT_data_tmp);
24.     if all_parameters_updated == true
25.          $LUT\_data = LUT\_data\_tmp$ ;
26.         reset_all_parameters_updated(LUT_data_tmp);
27.     end if
28. end while

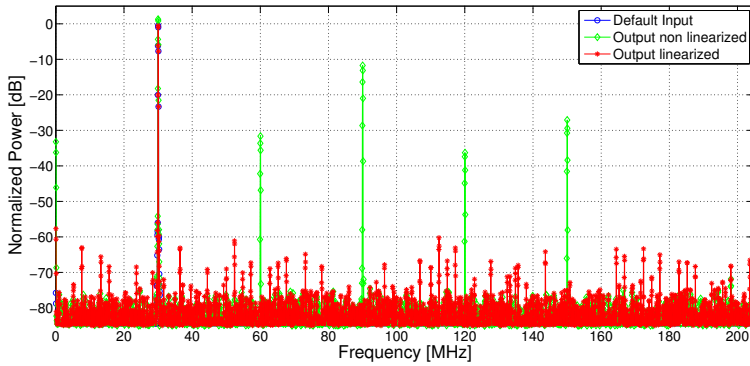
```

### 6.4.2 Results

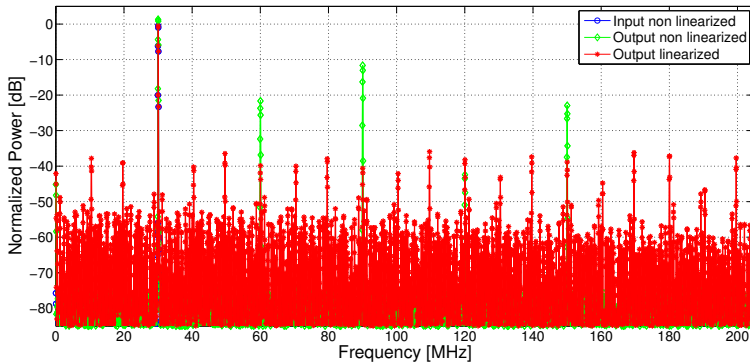
In this section results will be showed when simulating the LUT algorithm on the memoryless PA model and on the memory PA model from Section 6.1. Two types of signals will be used, a one-tone signal and a two-tone signal, like in the NMA simulations. The default design parameters for the algorithm are  $\mu = 2^{-3}$  and  $N = 256$ . The simulations are done with data that is quantized to have a 14 bit

representation at each calculation.

In Fig. 6.9 and 6.10 one can see the linearized result for the memoryless PA model and the memory PA model respectively. They both show the results when the used algorithm has entered a converged stage. The memoryless result shows a very good improvement in dBc compared to the default case, from -13.03 dBc to -59.70 dBc. As suspected the dBc result for the memory PA model is worse and a clearly higher noise floor is also visible. The method is as mentioned mainly designed to work for memoryless PA models. But yet it got a significant improvement with a default -12.94 dBc and a linearized result of -35.63 dBc, a performance that is in level with the NMA simulations.



**Figure 6.9:** A DFT of the linearization result when simulating a 30 MHz one-tone signal on the memoryless PA model.



**Figure 6.10:** A DFT of the linearization result when simulating a 30 MHz one-tone signal on the memory PA model.

### Memory Size

When increasing the size of the memory an improved performance is expected since the resolution will increase for the predistortion signal. In the case when

simulating towards the memoryless PA model this can clearly be seen in the values in Table 6.3. For the largest memory sizes the linearized results are almost in level with the added noise level of -80 dB. But for the case of the memory PA model the values saturate early and no further improvement in performance is visible when increasing the memory size. With the added memory effects on the PA model this method has a hard time to linearize since the distortion parameters are handled independently.

Memory size ( $N$ )	Memoryless PA model		Memory PA model	
	NMSE	dBc	NMSE	dBc
8	-18.98	-26.38	-16.00	-23.79
16	-25.54	-33.01	-17.66	-28.36
32	-30.37	-39.98	-21.03	-33.29
64	-35.49	-47.31	-23.57	-36.58
128	-41.91	-54.21	-23.26	-35.49
256	-46.08	-59.70	-24.36	-35.63
512	-48.67	-72.52	-24.72	-35.24
1024	-48.88	-73.26	-24.23	-35.10
2048	-48.89	-73.08	-24.49	-34.89

**Table 6.3:** Results for different memory sizes.

### Scale-factor

When decreasing the scale-factor  $\mu$  one could expect an improvement of the results. The reason for this is that a smaller scaling would mean a slower but more accurate accumulation to the searched predistortion signal. This is seen in Table 6.4 for both PA models, but the changes are not so drastic except for the first value. Bad result for  $\mu = 1$  are reasonable since it does not perform any scaling. Looking at the result for linearizing the memory PA model it is seen that this linearization method is limited. The memory effects bound the result to not improve further. Changing either  $N$  or  $\mu$  do not give the same effect as when linearizing the memoryless PA model. However, it still gets a significant improvement compared to the default input where no predistortion is performed.

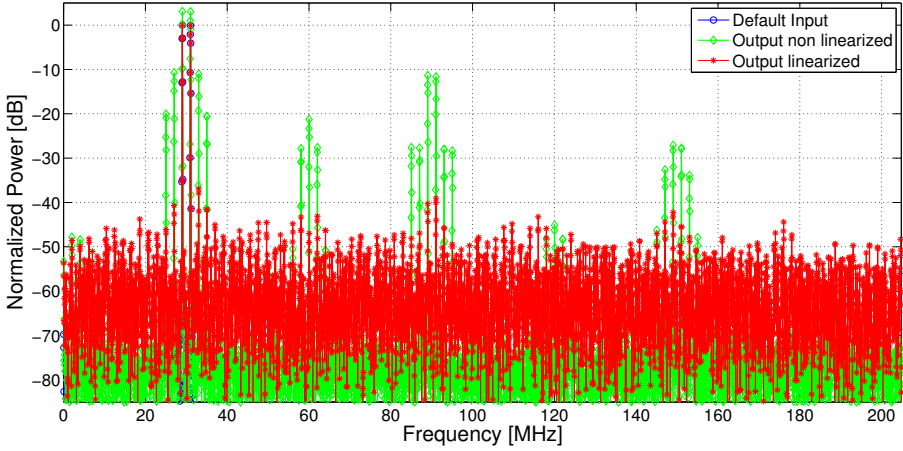
Scale-factor ( $\mu$ )	Memoryless PA model		Memory PA model	
	NMSE	dBc	NMSE	dBc
$2^{-0}$	-7.15	-20.24	-4.52	-19.76
$2^{-1}$	-42.37	-54.36	-21.99	-35.41
$2^{-2}$	-45.26	-58.95	-23.87	-35.50
$2^{-3}$	-46.08	-59.70	-24.36	-35.63
$2^{-4}$	-45.86	-60.42	-24.55	-35.72
$2^{-5}$	-44.62	-60.28	-24.66	-35.81
$2^{-6}$	-41.06	-60.70	-25.38	-36.51

**Table 6.4:** Results for different scale-factor.



### Two-tone Test

A two-tone test was also carried out with the memory PA model. The default dBc value is -13.73 dBc in the non linearized case. In Fig. 6.11 the linearization result is seen of -39.05 dBc. Comparing the dBc result with one-tone test they are about the same.



**Figure 6.11:** A DFT of the LUT linearization result when simulating a two-tone on the memory PA model using  $f_c = 30\text{MHz}$  and  $\Delta f = 1\text{MHz}$ .



# 7

---

## Implementation

This chapter presents the design of the implementations of the predistorter, both the model based version as well as the model-free one. It also shows their results with different choices of design parameters. All implementations are designed for the Xilinx FPGA that is mounted on the FOI MPX board. The FPGA has a limited set of hardware resources that can be utilized. A review of how much of these resources that has been used for each implementation will be presented. On the MPX board the DAC and ADC are also mounted. Important to note is that when producing a predistortion signal is that room has to be taken for adding the distortion part, meaning that when feeding the signal to the DAC some back-off has to be made to minimize the risk of overflow.

### 7.1 Base Implementation

The implementations were to be constructed together with a previous design from FOI. The purpose of the system is to transmit a pure sine wave that has a wide selectable range of frequencies. The following section will present how it is implemented.

#### 7.1.1 Direct Digital Synthesizer

A direct digital synthesizer (DDS) is a frequency synthesizer that can generate an arbitrary waveform with a chosen frequency. The goal of the DDS is to transmit a stream of digital values that represents the desired waveform. In this case a sinusoid of a wide frequency range is desired.

The concept of a DDS is not too complex, it is based on storing the sinusoid values in a LUT of size  $N$ . When storing the values of a sinusoid the fact that it is symmetric in two ways can be exploited and reducing the LUT size by four with

no performance loss. A pure sine wave is symmetric around the DC-point point as well as both sides of the two extreme values. When constructing the signal a phase accumulator is used and for each new sample a step of length  $\kappa_{step}$  is added to the phase accumulator. The modulo of the accumulator with respect to the size of the LUT is then used as the address pointer. The step size is calculated as

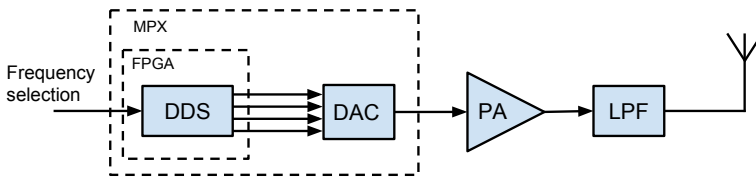
$$\kappa_{step} = \text{round}\left(N \frac{f_d}{f_s}\right) \quad (7.1)$$

where  $f_d$  is desired frequency,  $f_s$  is the sampling frequency and  $N$  is the size of the LUT. What can be pointed out is that the value  $\kappa_{step}$  has to be an integer since it is used for addressing. This makes a quantization of the  $f_d$  but the frequency resolution can be increased by increasing  $N$ .

The DAC used is designed for a fixed-point representation with a bit width of 14. Naturally the DDS is then also designed for a bit width of 14. The chosen number representation is two's complement.

### 7.1.2 System Overview

The only function implemented on the FPGA in the beginning was a DDS. It has four parallel outputs clocked at  $clk_{DDS} = 307.2$  MHz. All output samples have different phases compared to each other so that the concatenation of the samples gives a segment of a pure sinusoid. These signals are then transmitted from the FPGA to a DAC. The DAC has a clock frequency of four times the DDS output, making the analog output clocked at  $clk_{DAC} = 1228.8$  MHz. This limited the input to the PA to  $1228.8/2 = 614.4$  MHz. The frequency was selected with a PC through a USB interface. The base system is shown in Fig 7.1.



**Figure 7.1:** An overview of the base system.

As seen, no linearization technique is used so it is basically a signal generator. The DPD should be implemented after the DDS, using the four DDS outputs as its inputs so the data to the DAC will be predistorted.

### 7.1.3 Hardware Resources

The FPGA mainly consists of slices that builds up look-up tables and flipflop registers that combined implements a sequential logic net. In addition to slices the FPGA also consists of some dedicated hardware. Table 7.1 shows that there are

many hardware resources unused in the base implementation. The about 58,000 slices available are far from fully used in the design. The digital signal processing (DSP) slice and BlockRAM are dedicated hardware blocks on the FPGA which are used for constructing signal processing units and FIFO memories. These two types of hardware blocks will be used more in the implemented predistorter designs.

# used of	Slices			DSP	BlockRAM
	Logic	Memory	Register		
Base impl.	2,265	129	3,044	0	35
Available	58,880	58,880	58,880	640	244

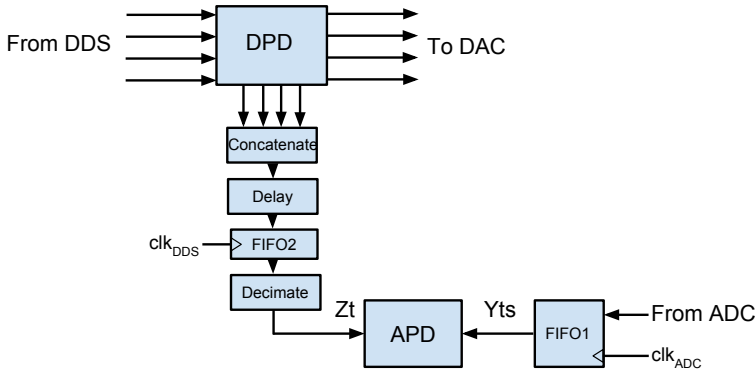
**Table 7.1:** Used hardware for the base implementation.

## 7.2 Adaptive Architecture

To use the adaptive predistortion a feedback had to be introduced. This meant that the system had to collect data from the ADC. The clock frequency of the ADC was three times lower than the DAC, i.e.  $clk_{ADC} = 1228.8/3 = 409.6$  MHz. The difference in clock speed had to be carefully considered in the design since the output of the DPD and the input from the ADC had to be compared for the adaptive algorithms. The solution was to use two FIFOs, see Fig 7.2. The ADC input was directly stored in the first one, FIFO1, using  $clk_{ADC}$  for sampling the data. FIFO2 was used to store the DPD output and was sampled with  $clk_{DDS}$ . However, the DPD data first had to be concatenated and delayed before stored in the FIFO2. A concatenation is needed to be able to manage the four parallel samples each clock cycle. The reason for the delay was to be able to compare the correct values for the cost function in the adaptive predistorter block. Since time is quantized in a digital system these correct values can only be sampled at a time that has the closest estimate to the true delay. The closest estimate was the number of DDS clock cycles it took for the signal to go from the DPD to the DAC then through the analog part of the system and later sampled and stored in FIFO1.

When both FIFOs were filled with a chosen number of samples the adaptive algorithm started its calculations. However, since FIFO2 contained more data some decimation was necessary. FIFO2 collected four data samples every  $clk_{DDS}$  whereas FIFO1 only collected one data sample every  $clk_{ADC}$ . This meant that FIFO2 contained  $\frac{4 \cdot clk_{DDS}}{1 \cdot clk_{ADC}} = 3$  times more samples than FIFO1. Because of this FIFO2 was decimated by three. The procedure of filling both the FIFOs with sampled data is continuously done so that the parameters in the DPD block can be updated. Since the total system has no external impacts a convergence of the best approximated inverse is to be expected, at this point of time no more parameter updates are needed.

The APD and DPD blocks are different for the NMA and LUT implementations.



**Figure 7.2:** The adaptive predistortion architecture. Within the APD block continues updates for the DPD parameters are made.

These will be discussed more detailed in the following sections.

### 7.2.1 Calibration Sensitivity

There are three major calibrations needed to be performed in the adaptive implementation related to the analog part of the system. The most critical one is the amount of delays to be performed within the delaybox seen in Fig. 7.2. It is affected by the propagation time through the PA, attenuator and the setup of cables besides the known time within the FPGA. Both implementations suffer a severe performance loss if not matched perfectly. The reason is comprehensible because the algorithms do not see the affect from the PA as expected. The other two needed calibrations are DC offset and scaling of the sampled values. The algorithms expect samples biased around zero, so to tune for this a calibrated constant parameter is subtracted from each of the sampled values.

Scaling of the sampled values is needed since the attenuator and the PA do not fully normalize the values and also the ADC and the DAC are operating with two different voltage ranges. This is solved by a calibrated constant parameter multiplied for each of the sampled values. All these calibrations add levels of complexity when debugging.

## 7.3 NMA Implementation

This section describes the procedure of the NMA model implementation. It includes theory and design choices as well as all the different hardware modules needed for building up the whole system.

### 7.3.1 Horner's Method

Horner's method is an algorithm for calculating polynomials that can reduce the number of multiplications and additions needed. This is useful on the FPGA

since every arithmetic operation consumes resources, especially multiplications. Given a polynomial,  $f(x)$ , the method starts by factoring out  $x$ . This is repeated until there is no more  $x$  to factorize, which gives the final form. The procedure is shown in (7.2) below:

$$\begin{aligned} f(x) &= a_0 + a_1x + a_2x^2 + \cdots + a_nx^n \\ &= a_0 + x(a_1 + a_2x + \cdots + a_nx^{n-1}) = \cdots \\ &= a_0 + x(a_1 + x(a_2 + \cdots (a_{n-1} + a_nx))). \end{aligned} \quad (7.2)$$

For example, a polynomial of order 3 the result will be  $f(x) = a_0 + x(a_1 + x(a_2 + a_3x))$  which uses 3 multiplications and 3 adders. Without Horner's method 6 multiplications and 3 adders are needed.

### 7.3.2 Number Representation

As mentioned in Section 7.1.1 the DDS has a 14-bit two's complement output. This is used as the input to the DPD system. It is represented as 1.13 to use the same conditions as in simulations. This means that 1 bit is used to represent the integer and 13 bits to represent the decimals. However, since the two's complement representation is used, the first bit will have a negative weight. The range for the DDS output is therefore  $-1 \leq x_{DDS} < 1$ . The same representation is used for the DPD output since the DAC also uses 14-bit two's complement.

To avoid problems with converting the numbers all the time, fixed-point two's complement representation is used for the coefficients as well. The number of bits used for the coefficients are 18 even though the accuracy from this is probably not needed, see Section 6.3.1. The main reason for this is to use the 25x18 multiplier of the DSP slice to its fullest. The coefficients are represented as 4.14. With the two's complement this gives  $-8 \leq a_i < 8$  and solutions with coefficients beyond these bounds will overflow.

### 7.3.3 DPD

The NMA model for the DPD used a polynomial of order five, excluding the fourth order as mentioned in Section 6.3.1. This gave the following function (excluding memory terms):

$$f(x) = a_1x + a_2x^2 + a_3x^3 + a_4x^5. \quad (7.3)$$

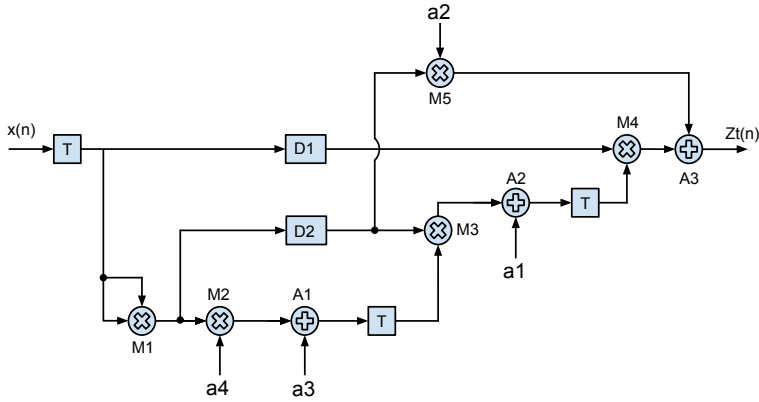
Using a modified version of Horner's method the function was rewritten as

$$f(x) = x(a_1 + x^2(a_3 + a_4x^2)) + a_2x^2. \quad (7.4)$$

This function can be seen as the DPD kernel function. The implementation of this function is shown in Fig. 7.3. As seen 5 multipliers and 3 adders were needed. The inputs to the multipliers has been truncated such that only 25x18 (or lower) multipliers are used. This means that every kernel uses five DSP slices. The

T blocks in the figure are single pipeline<sup>1</sup> stages used at the input and after the additions to prevent timing issues<sup>2</sup>. The multipliers have internal pipeline stages for the same reason. To make sure that the data arrives to the multipliers at the same time the pipeline delay blocks D1 and D2 are used. The delay D2 is equal to the time it takes for M2 to finish plus the one pipeline after A1. The delay of D1 is simply the time it takes for the lower branch to finish.

The values  $a_1$  to  $a_4$  are the coefficients for the kernel function. These coefficients are continuously updated by the APD unit. This will be discussed more detailed in the subsequent section.



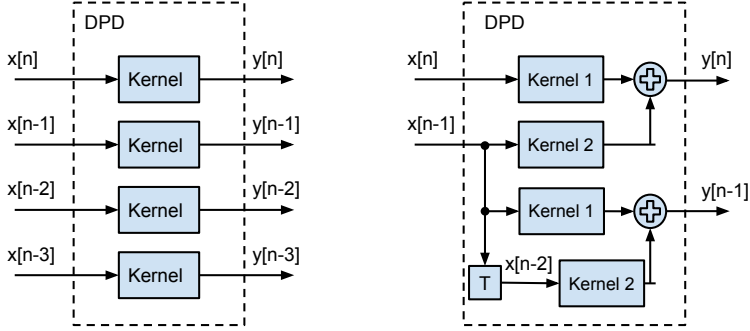
**Figure 7.3:** Kernel function implementation.

For every input to the DPD a kernel instance is needed, which means that at least four instances are necessary since the DPD has four inputs (see Fig. 7.4). However, for every memory depth in the model, additional 4 kernel instances are needed, each with its respective delayed input. These kernels are then added together to get the correct DPD output. This is shown in Fig. 7.4 with only 2 inputs and 1 memory for simplicity. Worth noting is that every memory has its own set of coefficients. For the given example of the model with 1 memory this means that a total of  $2 \times 4 = 8$  coefficients are needed.

<sup>1</sup>Block that delays the signal one clock cycle.

<sup>2</sup>Problem that occurs when the time through some combinational logic is larger than the clock period, i.e. the combinational logic do not have time to calculate its value before a new clock pulse occurs.





**Figure 7.4:** DPD block for a 4 input memoryless NMA model (left) and a 2 input NMA model with a single memory (right).

### 7.3.4 APD

The APD block calculates the coefficients for the DPD kernels. To do this the LMS algorithm was used. The NMA model was first rewritten as (6.4), i.e. the same linear regression model as used in the simulations.

The implementation of the LMS algorithm is shown in Fig. 7.5. Worth noting is that the design was chosen to work sequentially. This means that before a new input can be used in the system all the new coefficients needed to be calculated. The time between new inputs could potentially be a large number of clock cycles, depending on the memory depth. This was different from the DPD which worked in parallel, meaning that a new input sample was obtained every clock cycle.

Starting from the right in Fig. 7.5,  $Y_t$  and multiplications of  $Y_t$  gets stored in registers. These then get pipelined to new registers for every new input value. The regression vector  $\varphi(k)$  consists of these register values. The parameter vector  $\theta$  is created by the registers  $a_1, \dots, a_N$  in the upper left part of the figure. The vector multiplication,  $\varphi(k)^T \theta(k-1)$ , from (3.25) is then performed element by element by M1, using few resources. The control signal  $C_1$  makes sure that the correct values from the vectors are chosen.

The results are then added together in an accumulator register. The next part of the algorithm calculates the residual,  $\epsilon = Zt(k) - \varphi(k)^T \hat{\theta}$ , where  $Zt(k)$  is the input FIFO2. The residual is then multiplied to get  $\mu \times \epsilon$ . To save resources the multiplication is performed by right shifting the signal. One shift equals  $\mu = 0.5$ , two shifts equals  $\mu = 0.25$  and so on. The final multiplication was performed after this with the regression vector, once again one element at a time. The product was then added to the corresponding coefficient,  $a_i$ , to get the updated coefficient value. When the new coefficients,  $a_{temp}$ , were stored back in their registers a new  $Y_t$  input was obtained and the procedure started all over again.

### 7.3.5 Results

The two main considerations when testing the predistorter was what frequencies and what input power to use for the PA. The idea for the frequency choices was



nique are also shown in the table as well as the power of the fundamental tone at the output of the PA. The dBc is around -13 for all frequencies except 90 MHz where it is much lower, even lower than the requirement. The reason for this is that it, as mentioned before, only contains the second harmonic. Another observation is that the fundamental tone gets weaker for higher frequencies. This can be explained using the knowledge from Section 3.1.1, where it is shown that odd order terms will add power to the fundamental tone. Thus, the lower frequency used, the more odd order harmonics will be in-band and thus more power will be added to the fundamental tone.

The other consideration is the effect for different input power levels which will be tested in the coming section. All tests in this section were carried out with  $\tau = 3$  if nothing else is mentioned.

Freq.	dBc	Fund. (dBm)	Number of harmonics
25 MHz	-13.38	48.0	7
35 MHz	-12.75	47.4	5
45 MHz	-12.97	47.0	4
65 MHz	-13.44	46.6	3
90 MHz	-45.09	46.2	2

**Table 7.2:** Distortion values before using any linearization technique.

### Variations of $\mu$ and Frequency

Table 7.3 shows the results for the predistorter using different values of  $\mu$ . 90 MHz was excluded since it already met the linearity requirements. As seen, all cases improves the linearity compared to the values in Table 7.2. However, the cost for this is that power of the fundamental tone gets lower. The trade-off between dBc and power in the fundamental tone will be discussed later. Of all the cases  $\mu = 1$  gives the worst performance. By comparing the other two cases it seems like  $\mu = 0.25/4$  is the best one, at least regarding the dBc. It does, however, have a lower fundamental tone than the other cases.

For even lower  $\mu$  values the results are degraded.

Freq.	$\mu = 1$		$\mu = 0.25$		$\mu = 0.25/4$	
	dBc	Fund.(dBm)	dBc	Fund.(dBm)	dBc	Fund.(dBm)
25 MHz	-18.10	47.1	-20.05	47.1	-23.77	46.3
35 MHz	-18.04	46.2	-20.21	46.5	-20.89	45.8
45 MHz	-15.02	45.3	-15.53	46.0	-17.84	45.5
65 MHz	-32.76	45.0	-32.81	45.0	-36.48	44.6

**Table 7.3:** Linearized results for different  $\mu$  and frequencies.

### Power Levels

As mentioned before, the input power to the PA is one of the most important issues to consider when testing the predistorter. Therefore tests were carried out

with different input power levels to see the effects on the system. All tests were carried out with a 30 MHz input signal. The results are shown in Fig. 7.6. As seen, the drop in fundamental tone is almost equal for all inputs. The distortion shows best results for inputs between -9 dBm and -4 dBm.

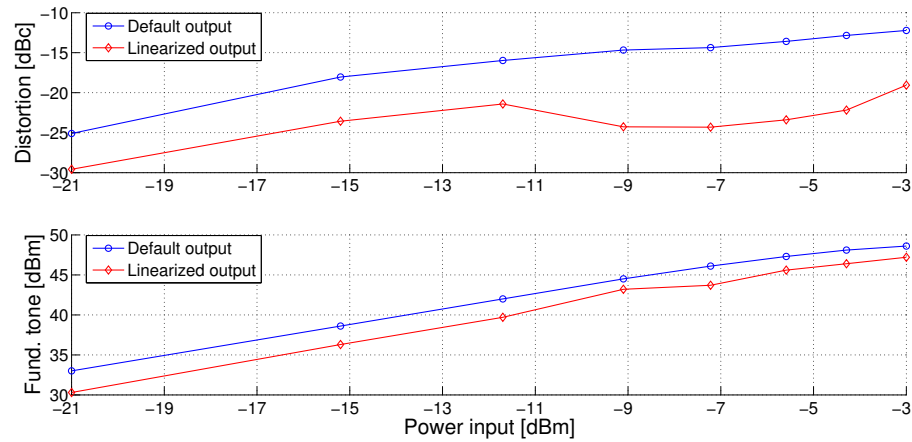


Figure 7.6: Results for varying power levels.

Memory Length

Tests were carried out to see how the performance varied with different memory lengths. This was performed with two different models, one with the fifth order term and one without it. This was carried out to see the impact of the fifth order term. The input frequency used this time was 30 MHz with a default dBc of -13.19 and output fundamental power of 47.6 dBm. The results are shown in Table 7.4. The first thing to note is that without any memory there is basically no improvement at all for both the models. The only impact the algorithm has in that case is to lower the input. By adding just one memory the dBc is improved by almost 10 dB. However, adding even more memory length does not give any noticeable better performance. Another observation is that the results are almost equally good using a model with the 5th order term as without it.

Memory length ( $\tau$ )	With 5th order		Without 5th order	
	dBc	Fund.(dBm)	dBc	Fund.(dBm)
0	-13.38	45.8	-12.61	46.5
1	-22.85	45.9	-22.05	46.0
2	-20.90	46.4	-19.75	46.0
3	-21.21	46.0	-22.95	46.0

Table 7.4: Linearized results for different memory lengths.

### 7.3.6 Hardware Resources

Table 7.5 gives a summary of the hardware resources used for the NMA model implementation. Compared to Table 7.1 more resources are needed, which is expected since it is a larger design. Worth noting is that these values are calculated with a NMA model with a memory depth of three. Using less memory will of course use less resources. Especially the DSP slices which will be decreased by 20 for every memory removed. The increase of used blockrams are because of the two introduced FIFO memories.

# used of	Slices			DSP	BlockRAM
	Logic	Memory	Register		
NMA impl.	5,473	1,483	9,138	85	67
Available	58,880	58,880	58,880	640	244

**Table 7.5:** Used hardware for the NMA model implementation.

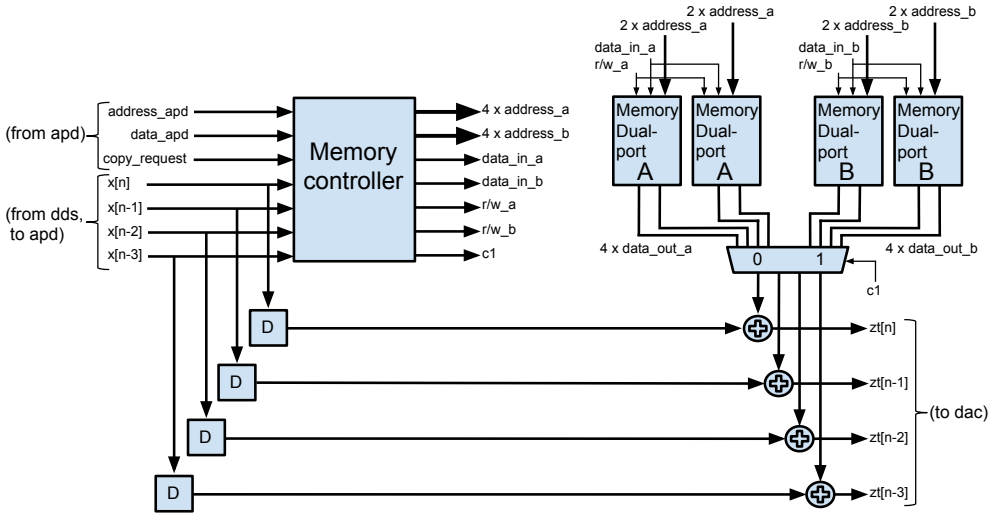
## 7.4 LUT Implementation

This section describes the procedure of the model-free implementation. It shows how the simulated algorithm has been mapped to hardware. What is aimed to achieve is the same as in the simulations, i.e. distortion parameters in a LUT that together constructs the predistortion signal. This LUT is the outcome of one adaption process of the LUT implementation, specified for the current frequency.

### 7.4.1 DPD

Figure 7.7 is a system overview of the DPD implementation that is designed from the algorithm in Section 6.4.1. The algorithm itself is simple, the design is mostly shaped by the need of controlling the memory setup, hence the memory controller. Adjustments according to the base implementation design had to be made. The continuous input of four samples from the DDS introduced the need for reading four distortion parameters each clock cycle and since transmitting a signal is a time critical operation, it is not desired to be put to a hold. However, the total system is an adaptable solution so there is a need to update the contents of the memories. It is solved by having two equal memory setups *A* and *B*, one that is read from and the other that can be updated and written to. Each of the setup *A* and *B* consist of two identical dual-read memories making it possible to read four values each clock cycle. When a satisfied solution has been reached from the APD there is no need for two memory setups. But when extracting a solution the memory controller will toggle between *A* and *B* by the control signal *c1*.

The pipeline delay boxes *D* are needed to compensate for the time it takes to fetch the corresponding distortion parameter in the memory. The four addresses produced for the current memory setup *A* or *B* are constructed by concatenating the sign derivative and a truncation of the DDS sample. The sign derivative put



**Figure 7.7:** System overview of the DPD unit with two memory setups A and B with two dual-read memories each.

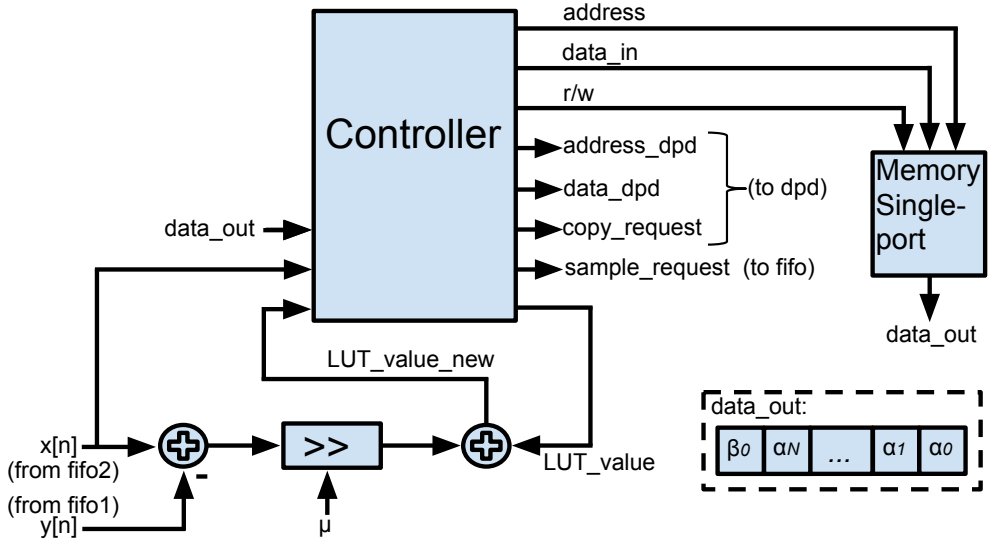
as MSB divides a memory into two parts. The four sign derivatives is simply a comparison between two consecutive samples. The  $X[n]$  sample is pipelined so that the derivative for  $X[n-3]$  can be calculated. The size of the address truncation is in relation to the size of the memories. The pure sinusoid generated from the DDS is biased around zero making it possible to have a direct mapping fully utilizing the whole memory if sinusoid has full swing. Further on the four address outputs from the memory controller are inputs to the correct address port in correct memory setup, matching the routing so the four *data\_out* will match the four delayed samples.

The decision of toggling between the memory setup A and B comes from the APD unit that works as a master. The *copy\_request* starts the process of feeding the continuous stream of *data\_apd* filling up the two memories currently not in use for reading. The stream stops when the *address\_apd* has made a full circle so the whole memory has been updated. This makes a major disadvantage for the method because it will be time consuming replacing the whole memory if a large memory is used. This system is intended to be used as a jamming system, the ability of switching frequency fast is limited by the time it takes to replace all the distortion parameters in the memory. But in return the only functional operation that is done is the additions made before the output to DAC.

### 7.4.2 APD

A system overview of the design for the APD can be seen in Fig. 7.8 and just as the DPD design it is mostly a matter of controlling the memory. The APD unit is not a time critical part and is designed as a master over the total system. From the APD unit the control signals *copy\_request* and *sample\_request* are transferred

to the DPD unit and the two FIFO memories respectively. The APD unit can then work in its own pace by sending a *sample\_request* pulse for getting two new samples  $x[n]$  and  $y[n]$ . With the two samples a possible new *LUT\_value* can be calculated to be stored in the single memory. The decision of storing or not is made if it has been updated since last copying to the DPD, from here on called a window. What is done through the functional path is accumulation of the currently stored predistortion parameter with a scaled error. The scaling is implemented by a certain number of shifts according to  $\mu$  instead of introducing a costly multiplier.



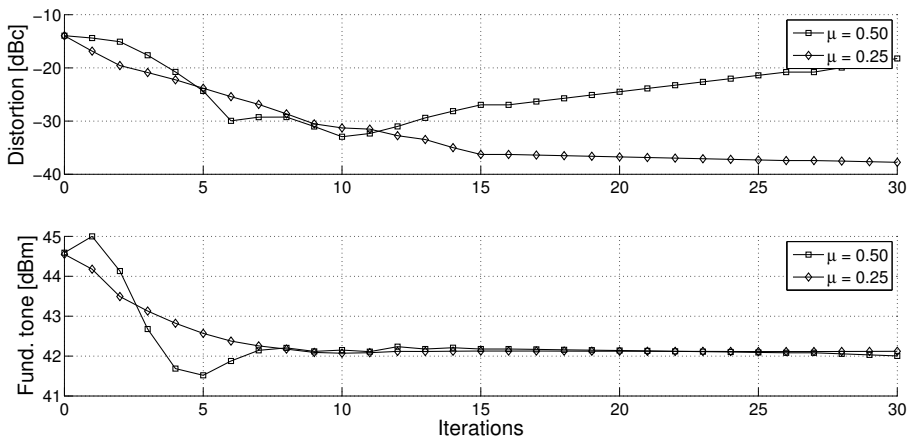
**Figure 7.8:** System overview of the APD unit with one single-read memory. The dashed box shows the partitioning of the content for each cell in the memory.

In Fig. 7.8 the dashed box shows the partitioning for a cell in the memory. The  $\alpha_i$  bits makes up the stored *LUT\_value* also called distortion parameter. The  $\beta_0$  is the bit that tells if the stored value has already been updated within the current window. The change of a window is done when a pulse *copy\_request* has been received, meaning a start of copying the whole memory to the DPD unit. During this time the APD is fully occupied by this task so no requests for new samples are done. The update bits in the local memory are being reseted during the copying process to the DPD. The question of when to send a *copy\_request* is directly related to how many addresses in the memory that should have a set  $\beta_0$ . This varies with the chosen frequency and amplitude. What is desired by the algorithm is to send a *copy\_request* when there is no more distortion parameters left to update. The solution is to have an internal counter for the number of set  $\beta_0$  and when this counter has been inactive for a longer time a *copy\_request* is sent. This longer time together with copying whole memories will make this method

not so time effective. But if it gives a good result and convergence time is not highly prioritized it may be a good trade-off anyway.

### 7.4.3 Results

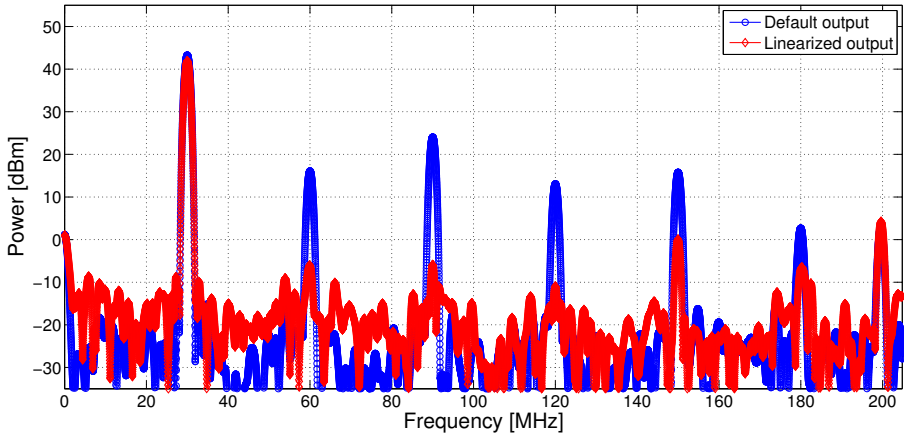
Similar tests that has been carried out on the NMA model implementation has also been carried out on the LUT implementation. The scale-factor is the LUT implementations variant of step size in the NMA implementation. What is desired is a scale-factor that will give a converged and stable system. Figure 7.9 shows the measured distortion in dBc for two sets of scale-factors  $\mu$ . The used frequency is 30 MHz, default input power is -9 dBm and the LUT memory size is 256. For  $\mu = 0.25$  the dBc value shows convergence after 15 iterations. The case for  $\mu = 0.50$  shows a clear system instability. Continuing with iterations for this case would give a worse result than the base implementation (Iterations = 0). Choosing an even lower  $\mu$  would give a slower convergence but not a better result. Looking at the lower plot in Fig. 7.9 the output power of the fundamental tone is visible, for both cases it quickly decreases. When linearized it is less affected by its harmonics due to the nonlinearities. Meaning a drop of the output power is a direct consequence when linearizing.



**Figure 7.9:** Measurements of dBc plotted against iterations of the algorithm (upper), results for two different values of scale-factors  $\mu$ . Value of the power for the fundamental tone (lower).

The result for the case with  $\mu = 0.25$  is plotted as a DFT in Fig. 7.10. The default output with no iterations made and the converged linearized output is shown. The harmonics with lowest frequency are the ones that have been suppressed the most, and the dBc is bound by the aliased frequency bin to -37.7 dBc. Some spurs have higher power levels than the most suppressed harmonic, a sign that no more attenuation of these frequencies is to be expected with this implementation. The mentioned settings of frequency, power level, memory size and a  $\mu = 0.25$  will be the default setting for further tests if nothing else is noted.





**Figure 7.10:** A DFT plot showing the linearized result compared to the default result.

### Frequency

Choosing frequency impacts the dBc performance. In Table 7.6 the result for different frequencies are shown. The three lowest frequencies gives the worst result, mainly because they all have third and fifth harmonic in band. But a higher frequency also means a harder task to extract a good signal for the predistortion LUT memory. This can be a reason for the divergent result at 65 MHz.

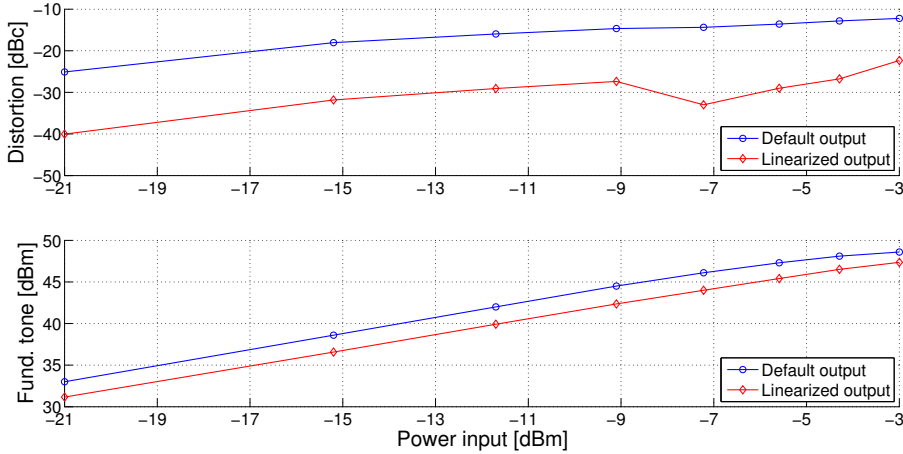
Frequency	Default		Linearized	
	dBc	Fund.(dBm)	dBc	Fund.(dBm)
25 MHz	-13.30	47.97	-28.67	45.41
30 MHz	-12.94	47.62	-29.82	45.31
35 MHz	-13.64	47.39	-28.32	45.33
45 MHz	-13.26	47.05	-42.77	45.76
65 MHz	-14.29	45.45	-34.31	45.37

**Table 7.6:** Results when varying the frequency.

### Power Levels

As mentioned before the performance measured in dBc will be degraded if the input power is increased and the power of the fundamental tone will then start to saturate. In Fig. 7.11 default measurements of this can be seen as well as the result of the linearization method to corresponding power level. Comparing the dBc values one can see they have similar incline, but for the highest values the method has a hard time linearizing thus a steeper incline is visible. In the lower graph in Fig. 7.11 the power of the fundamental tone is plotted. Also here the observation can be made that the incline of the values is similar but for higher values the saturation is noticeable for the default case. The drop of dBc between

the input powers -9 and -7 dBm are quite substantial, no good explanation for this can be given more than that the circumstances was very favorably during this test.



**Figure 7.11:** Results for varying power levels.

## LUT Size

When simulating different LUT sizes in Section 6.4.2 there was a clear result of better performance with bigger memory size if the PA model had no memory effects. This is expected since it gives a higher resolution in the LUT. But if the PA model had memory effects there was not a big difference between different LUT sizes.

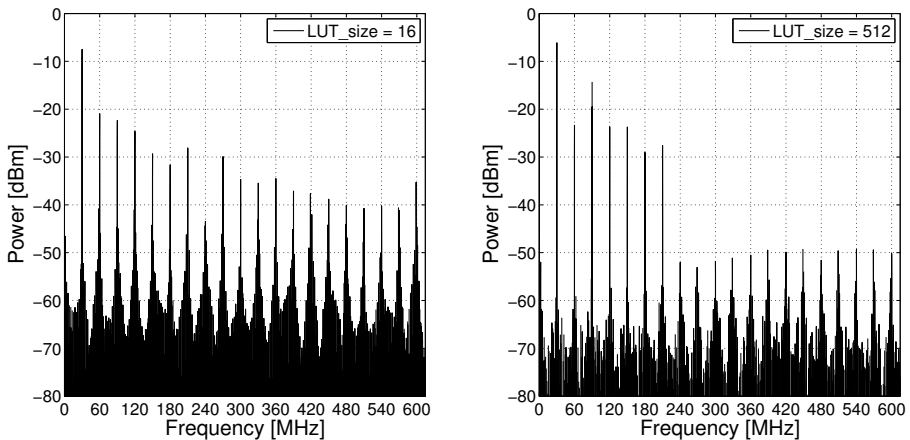
In Table 7.7 the result is presented when varying the LUT size in the implementation. The corresponding values for the nonlinearized output was -13.30 dBc and 47.97 dBm in the fundamental tone. As seen there is not a big difference in performance between the different LUT sizes. The hypothesis for the small differences is memory effects of the PA. This LUT linearizing technique seems to be bound by this and no further improvements can be seen with higher  $N$  nor smaller  $\mu$ . The hypothesis is supported by the result from simulations and implementation. However, how the PA memory effects vary with different conditions is not fully analyzed, meaning that this hypothesis is not resolved in this thesis.

Looking closer at the smallest and biggest memory sizes ( $N=16$  and  $N=512$ ) one can see that the extracted predistortion signals differ, seen in Fig. 7.12. The predistortion signal is constructed by the four parallel signals which are sent to the DAC. This means that the Nyquist frequency will be three times higher than for the ADC, 614.4 MHz compared to 204.8 MHz. Both signals are compound by frequencies corresponding to the harmonics of the fundamental tone 30 MHz. It is clear that the solutions are not the same and for  $N = 16$  the signal lacks some clarity. As can be seen in the plot there is more power in frequencies above the ADC Nyquist frequency. For  $N = 512$  the higher harmonics has less power.

Memory size $N$	Linearized	
	dBc	Fund.(dBm)
16	-26.50	45.29
32	-28.99	45.37
64	-27.82	45.36
128	-29.82	45.31
256	-29.56	45.50
512	-28.44	45.34

**Table 7.7:** Results for different memory sizes.

This is good since the constructed signal is created by sampled data, so it should not be constructed by such high frequencies. Also with bigger LUT size a lower quantization noise is noticeable. These results shows that the extracted solutions differ from each other, but comparing the resulting dBc the difference is only 2 dB.



**Figure 7.12:** Comparison between two different predistortion signals with different memory sizes,  $N = 16$  (left) and  $N = 512$  (right).

#### 7.4.4 Hardware Resources

Table 7.8 summarizes the used hardware for two LUT implementations. One implementation with a memory size of 32 and the other with a memory size of 512. Compared to Table 7.1 some more slices and blockrams has been used. The slices used for logic do not differ so much between the two implementations and no DSP block is needed for this method. However, there is obviously a difference in used slices when constructing memories. Altogether there are plenty of unused resources on the FPGA to construct cooperating designs.

# used of	Slices			DSP	BlockRAM
	Logic	Memory	Register		
LUT impl. 1	3,558	827	5,592	0	73
LUT impl. 2	3,713	1,656	5,934	0	74
Available	58,880	58,880	58,880	640	244

**Table 7.8:** Used hardware for two LUT implementation, LUT impl. 1 with memory size 32 and LUT impl. 2 with memory size 512.

# 8

---

## Conclusions and Future Work

### 8.1 Conclusions

In this thesis it has been showed that strongly attenuated harmonics can be achieved when reducing the input power together with the proposed linearization techniques. The presented result that showed the best result had a fundamental tone of about 42 dBm and the distortion was measured to -38 dBc, an improvement of 20 dB from the nonlinearized setup. However, this result does not hold for the first stated requirements of -40 dBc and maximized output. At the time of setting the requirements not much was known of the power and linearity trade-off, neither the expected performance when using such wide bandwidth. With present knowledge the first stated requirements might have been a bit excessive. No similar work has been found to confirm this statement, the theoretical performance limit is yet unknown. Jamming is a defense related topic so similar work is hard to come across.

What is mostly done in previous scientific work when it comes to digital predistortion, is to linearize within a small bandwidth. Outside this bandwidth of about a couple of 10 MHz an analog filter has attenuated all the harmonics. A modulated signal type has been used and attenuation of the intermodulation distortion has been in focus. But in this thesis the focus has been on using a one-tone signal and attenuate the harmonics over a wide bandwidth, a form of wideband predistortion. Improvement of linearization has been showed for both implemented designs. However, the model based implementation has showed less improvement compared to the model-free version. This conclusion shows evidence that the used nonlinear model might not work too good for such a wide bandwidth. The digital predistortion implementations that were designed in this thesis turned out to give a more linear output of the PA, by attenuating the harmonics. How-

ever, if one strives for having a maximized output power for the selected frequency it is not helpful to linearize the PA. A direct consequence of linearizing the PA was that the fundamental tone obtained a reduced output power since the nonlinearities of the harmonics did no longer add power to the fundamental tone. If one strives for attenuating the harmonics to its fullest a trade-off can be made by reducing the power of the fundamental tone.

During this thesis only behavioral modeling has been used and no extensive information about the complex internal setup of the PA design has been known. The shortage of this knowledge contributes to the lack of presented information about performance limitation and the sources of the PA nonlinearities and memory effects. What should be noted is that the sources of nonlinearities are not only derived from the PA, for example the ADC and DAC are part of the total system that is linearized. However, the assumption that it is mainly the PA that stands for the nonlinearities has not been questioned during the thesis.

The results between simulations and implementation are quite different. An explanation for this is that the used behavioral model is unable to capture the true behavior of the PA. This was expected, but nevertheless the PA model fulfilled the purpose of verifying that a nonlinear system could be linearized with the designed implementations.

The tests that have been performed in this thesis have used one specific PA. But measurements were also done on another class AB PA. The implemented predistortion techniques showed similar results when linearizing this PA.

### 8.1.1 Implementation

Both the implementations showed a decrease in performance compared to the simulations. One explanation for this has been given but there are some further uncertainties regarding the implementation of the adaptive part of the systems. The calibration sensitivity is of interest, the choice of design parameter for delay and thus sample correlation has showed large impact on the result. Varying performance was noticeable when fine adjusting the calibration parameter for the correlation. There was a clear best choice for this calibration parameter, meaning that neither increasing or decreasing the value of the parameter would give a better result for the current setup.

This calibration was as mentioned before limited by the sampling frequency of the ADC. Decimation of the DAC data was needed for adjusting to this limitation, therefore a loss in resolution. Reducing the resolution is not desired, because a predistortion signal that is meant to fully cancel PA nonlinearities has to be accurate to get the best result. Another evidence supporting this issue was that varying performance was also noticeable when having slightly different measurement setups, like cables with varying length or other analog filters. This is related because the propagation delay of the calibration path is then changed and could possibly have another optimum calibration. What ideally would have been desired is to have infinite sampling frequency. At least as high sampling frequency of the ADC as for the DAC so no decimation would have been needed.

For the model implementation it can be seen in the results that it does not seem to

matter if one uses one or three memories. Also the effect of the fifth order is not as strong as expected. A hypothesis for this may be that with the model implementation errors from varying sources could be enhanced from all the exponential calculations. In contrast, the model-free implementation includes less calculations and a simpler design. This might be a reason for why it gives a better result in nearly all types of measurements. Adding to the differences the model-free implementation has a higher noise floor on the predistortion signal compared to the model implementation, but it does not seem to suffer the ability to attenuate the harmonics. In the model implementation the parameters are more dependent since they all are derived from the same sampled data, compared to the model-free implementation where sampled data only affects one parameter at a time. This restriction for the model implementation might be a reason why it has a worse result compared to the model-free implementation.

Focusing on the capability of jamming, the two methods need to be able to change parameters for different frequencies at a fast pace. To not be limited by the adaption time when used in real time, all the adaption calibration can be initiated at start up, storing banks of parameters that are adjusted for small ranges of frequencies. Comparing the two methods the model-based implementation has a clear advantage because typically it has far less parameters that are needed to be stored in the bank. It also needs fewer transfers when replacing the current parameter setup to perform a frequency change.

## 8.2 Future Work

To have a complete jamming system one needs to develop an extended design, the ability for automatic frequency hopping has to be added. With a complete system an investigation can be done on how fast frequency hopping the implemented designs can allow. To exclude the need of manual adjustment an automatic calibration can be added to fine tune the analog setup. An ADC with higher sampling frequency can be used to investigate its effect, a higher bandwidth can then be utilized. More extensive tests on different PAs can be done to learn more about the limitations of this type of nonlinear system.

To further reduce the used hardware an optimization of the design can be made. The used hardware can be shared more between calculations, the used area would then be reduced when less hardware is needed. Other types of nonlinear models for the predistorter can be implemented to see if different results are achieved, also other linearization techniques can be tested. The simulations showed promising results for two-tone signals, implementing other types of input signals can tell more about the generality of the implemented designs. This would result in more hardware, in the case of a two-tone signal a second DDS would be added.





---

## Bibliography

- Anritsu. Adjacent channel power ratio (acpr), 2001. URL <http://downloadfiles.anritsu.com/Files/en-US/Application-Notes/Application-Note/11410-00264.pdf>. Cited on page 10.
- L. Anttila. *Digital Front-End Signal Processing with Widely-Linear Signal Models in Radio Devices*. Tampere University of Technology, 2011. ISBN 9789521526510. URL <http://URN.fi/URN:ISBN:978-952-15-2978-8>. Cited on page 10.
- S.C. Cripps. *RF Power Amplifiers for Wireless Communications, Second Edition (Artech House Microwave Library (Hardcover))*. Artech House, Inc., Norwood, MA, USA, 2006. ISBN 1596930187. Cited on page 29.
- L. Ding and G.T. Zhou. Effects of even-order nonlinear terms on power amplifier modeling and predistortion linearization. *IEEE T. Vehicular Technology*, 53: 156–162, 2004. Cited on page 17.
- G. Forsling and M. Neymark. *Matematisk Analys: en Variabel*. Liber, 2011. ISBN 9789147100231. URL <http://books.google.se/books?id=XwSetgAACAAJ>. Cited on page 20.
- F. Gustafsson, L. Ljung, and M. Millnert. *Signal Processing*. Utbildningshuset/Studentlitteratur, 2010. ISBN 9789144058351. URL <http://books.google.se/books?id=UDlMygAACAAJ>. Cited on page 25.
- Y. Jung. *Estimation of Inverse Models Applied to Power Amplifier Predistortion*. Linköping University Electronic Press, 2013. ISBN 9789175195711. URL <http://urn.kb.se/resolve?urn=urn:nbn:se:liu:diva-97355>. Cited on page 17.
- L. Ljung, editor. *System Identification (2Nd Ed.): Theory for the User*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 1999. ISBN 0-13-656695-2. Cited on page 24.
- F.L. Luo. *Digital Front-End in Wireless Communications and Broadcasting: Circuits and Signal Processing*. Cambridge University Press, 2011.

- ISBN 9781139499019. URL [http://books.google.se/books?id=Tu\\_c9XmSD1UC](http://books.google.se/books?id=Tu_c9XmSD1UC). Cited on pages 21, 31, 32, and 39.
- G. Mingming, N. Jingchang, and W. Surina. A novel power amplifier behavior modeling based on rbf neural network with chaos particle swarm optimization algorithm. *JCP*, 9(5):1138–1143, 2014. Cited on page 23.
- M.R. Moazzam and C.S. Aitchison. Low third order intermodulation amplifier with harmonic feedback circuitry. 1996. URL <https://login.e.bibl.liu.se/login?url=http://search.ebscohost.com/login.aspx?direct=true&db=edselc&AN=edselc.2-52.0-0029705507&site=eds-live>. Cited on page 30.
- P.L.G. Pinal, G.M. López, and E.B. Albertí. *Multi Look-Up Table Digital Predistortion for RF Power Amplifier Linearization*. 2007. ISBN 9788469147481. URL <http://books.google.se/books?id=o7djMwEACAAJ>. Cited on pages 23, 28, 29, and 30.
- J.M. Rabaey, A.P. Chandrakasan, and B. Nikolic. *Digital Integrated Circuits : a Design Perspective*. Prentice Hall electronics and VLSI series. Pearson Education, 2 edition, 2003. ISBN 0130909963. Cited on page 40.
- B. Razavi. *RF Microelectronics*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1998. ISBN 0-13-887571-5. Cited on page 11.
- A. A. M. Saleh. Frequency-independent and frequency-dependent nonlinear models of TWT amplifiers. *IEEE Transactions on Communications*, 29:1715–1720, 1981. Cited on page 19.
- D. Schreurs, M. O'Droma, A.A. Goacher, and M. Gadringer. *RF Power Amplifier Behavioral Modeling*. Cambridge University Press, New York, NY, USA, 1st edition, 2008. ISBN 0521881730, 9780521881739. Cited on pages 19, 20, and 24.
- S. Söderkvist and L.E. Ahnell. *Tidsdiskreta Signaler och System*. Erik Larsson AB, 1994. Cited on pages 16 and 29.
- I. Teikari. *Digital Predistortion Linearization Methods for RF Power Amplifiers*. TKK dissertations. Helsinki University of Technology, 2008. ISBN 9789512295456. URL [http://books.google.se/books?id=\\_1FDQwAACAAJ](http://books.google.se/books?id=_1FDQwAACAAJ). Cited on page 17.
- V. Volterra and E. Whittaker. *Theory of Functionals and of Integral and Integro-Differential Equations*. New York : Dover, 1959, 1959. ISBN 9900794508. URL <https://login.e.bibl.liu.se/login?url=http://search.ebscohost.com/login.aspx?direct=true&db=cat00115a&AN=lkp.13472&site=eds-live>. Cited on page 20.

## Upphovsrätt

Detta dokument hålls tillgängligt på Internet — eller dess framtida ersättare — under 25 år från publiceringsdatum under förutsättning att inga extraordinära omständigheter uppstår.

Tillgång till dokumentet innebär tillstånd för var och en att läsa, ladda ner, skriva ut enstaka kopior för enskilt bruk och att använda det oförändrat för icke-kommersiell forskning och för undervisning. Överföring av upphovsrätten vid en senare tidpunkt kan inte upphäva detta tillstånd. All annan användning av dokumentet kräver upphovsmannens medgivande. För att garantera äktheten, säkerheten och tillgängligheten finns det lösningar av teknisk och administrativ art.

Upphovsmannens ideella rätt innefattar rätt att bli nämnd som upphovsman i den omfattning som god sed kräver vid användning av dokumentet på ovan beskrivna sätt samt skydd mot att dokumentet ändras eller presenteras i sådan form eller i sådant sammanhang som är kränkande för upphovsmannens litterära eller konstnärliga anseende eller egenart.

För ytterligare information om Linköping University Electronic Press se förlagets hemsida <http://www.ep.liu.se/>

## Copyright

The publishers will keep this document online on the Internet — or its possible replacement — for a period of 25 years from the date of publication barring exceptional circumstances.

The online availability of the document implies a permanent permission for anyone to read, to download, to print out single copies for his/her own use and to use it unchanged for any non-commercial research and educational purpose. Subsequent transfers of copyright cannot revoke this permission. All other uses of the document are conditional on the consent of the copyright owner. The publisher has taken technical and administrative measures to assure authenticity, security and accessibility.

According to intellectual property law the author has the right to be mentioned when his/her work is accessed as described above and to be protected against infringement.

For additional information about the Linköping University Electronic Press and its procedures for publication and for assurance of document integrity, please refer to its www home page: <http://www.ep.liu.se/>