



UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Escola d'Enginyeria de Telecomunicació
i Aeroespacial de Castelldefels



Master's degree in Applications
and Technologies for
Unmanned Aircraft Systems

MASTER THESIS

TITLE: Automatic Transmit Power Control for Power Efficient Communications in UAS

MASTER DEGREE: Master's degree in Applications and Technologies for Unmanned Aircraft Systems (Drones) (MED)

AUTHOR: Wantao Li

ADVISOR: Pere L. Gilabert

DATE: June 7th, 2020

Abstract

Nowadays, unmanned aerial systems (UAS) have become one of the most popular tools that can be used in commercial, scientific, agricultural and military applications. As drones become faster, smaller and cheaper, with the ability to add payloads, the usage of the drone can be versatile. In most of the cases, unmanned aerial systems are equipped with a wireless communication system to establish a link with the ground control station to transfer the control commands, video stream, and payload data. However, with the limited on-board calculation resources in the UAS, and the growing size and volume of the payload data, computational complex signal processing such as deep learning cannot be easily done on the drone. Hence, in many drone applications, the UAS is just a tool for capturing and storing data, and then the data is post-processed off-line in a more powerful computing device. The other solution is to stream payload data to the ground control station (GCS) and let the powerful computer on the ground station to handle these data in real-time. With the development of communication techniques such as orthogonal frequency-division multiplexing (OFDM) and multiple-input multiple-output (MIMO) transmissions, it is possible to increase the spectral efficiency over large bandwidths and consequently achieve high transmission rates. However, the drone and the communication system are usually being designed separately, which means that regardless of the situation of the drone, the communication system is working independently to provide the data link. Consequently, by taking into account the position of the drone, the communication system has some room to optimize the link budget efficiency.

In this master thesis, a power-efficient wireless communication downlink for UAS has been designed. It is achieved by developing an automatic transmit power control system and a custom OFDM communication system. The work has been divided into three parts: research of the drone communication system, an optimized communication system design and finally, FPGA implementation.

In the first part, an overview on commercial drone communication schemes is presented and discussed. The advantages and disadvantages shown are the source of inspiration for improvement. With these ideas, an optimized scheme is presented. In the second part, an automatic transmit power control system for UAS wireless communication and a power-efficient OFDM downlink scheme are proposed. The automatic transmit power control system can estimate the required power level by the relative position between the drone and the GCS and then inform the system to adjust the power amplifier (PA) gain and power supply settings. To obtain high power efficiency for different output power levels, a searching strategy has been applied to the PA testbed to find out the best voltage supply and gain configurations. Besides, the OFDM signal generation developed in Python can encode data bytes to the baseband signal for testing purpose. Digital predistortion (DPD) linearization has been included in the transmitter's design to guarantee the signal linearity. In the third part, two core algorithms: IFFT and LUT-based DPD, have been implemented

in the FPGA platform to meet the real-time and high-speed I/O requirements. By using the high-level synthesis design process provided by Xilinx Corp, the algorithms are implemented as reusable IP blocks. The conclusion of the project is given in the end, including the summary of the proposed drone communication system and envisioning possible future lines of research.

CONTENTS

CHAPTER 1. INTRODUCTION.....	1
Motivation.....	1
Objectives	2
Technologies involved.....	2
Planning	3
CHAPTER 2. DRONE COMMUNICATION SYSTEM	4
2.1. Targeting UAS.....	4
2.2. Existing solutions.....	5
2.3. Optimized transmitter architecture.....	7
CHAPTER 3. SYSTEM DESIGN.....	9
3.1. Automatic transmit power control.....	9
3.1.1. Kalman filter for estimating the RX power	10
3.1.2. Simulation of the power and distance estimation in Python	13
3.1.3. TX power control logic	15
3.2. Design of an OFDM system	16
3.2.1. Encoder	17
3.2.2. QAM mapper	17
3.2.3. Pilots and serial to parallel conversion	18
3.2.4. IFFT	20
3.2.5. Cyclic prefix	20
3.2.6. OFDM baseband data samples.....	21
3.2.7. Channel simulation	24
3.2.8. Synchronization	26
3.2.9. Channel estimation	28
3.3. Techniques to improve power efficiency.....	30
3.3.1. PA bias point selection and operating classes	32
3.3.2. Overview of commercial power amplifiers.....	33
3.3.3. Digital predistortion techniques	34
3.3.4. The power amplification testbed.....	38
3.3.5. Searching for the optimal PA configuration	39
3.3.6. Perform DPD with the optimal settings.....	42
CHAPTER 4. FPGA IMPLEMENTATION.....	44

4.1. FPGA and Zynq SoC introduction	44
4.1.1. Zynq APSoC architecture	45
4.1.2. Zynq-based design methodology and practice	46
4.1.3. Useful PL resources	50
4.1.4. The Zedboard for experimental validation	52
4.2. High-Level synthesis.....	53
4.2.1. Fixed-point support	53
4.2.2. Directives for optimization	55
4.2.3. Generating in-chip interfaces.....	56
4.3. IFFT implementation	56
4.3.1. FFT IP from the HLS library.....	56
4.3.2. Evaluate the IFFT IP	57
4.4. DPD implementation	58
4.4.1. Architectures of implementing DPD in FPGA.....	58
4.4.2. LUTs extraction	60
4.4.3. Indexing logic and simulation	61
4.4.4. Synthesis and performance analysis.....	62
4.4.5. Final implementation	63
 CHAPTER 5. CONCLUSIONS AND FUTURE WORK	 66
 REFERENCES.....	 68

LIST OF FIGURES

Fig. 1.1 Kanban board schedule.....	3
Fig. 2.1 Spectrum of the DJI Lightbridge signal.....	6
Fig. 2.2 Block diagram of the drone transmitter and GCS receiver.	7
Fig. 2.3 Static TX power vs dynamic TX power.	8
Fig. 3.1 FSPL vs distance.....	12
Fig. 3.2 EKF RX power and distance vs time	14
Fig. 3.3 EKF RX power and distance vs time with obstacles.....	14
Fig. 3.4 EKF RX power and distance vs time with obstacle and go back	15
Fig. 3.5 Automatic power control logic.....	15
Fig. 3.6 The simulated control of TX power	16
Fig. 3.7 OFDM full implementation diagram [10]	16
Fig. 3.8 OFDM baseband generator	17
Fig. 3.9 QAM-64 gray coding constellation.....	17
Fig. 3.10 Sample image and QAM symbol distribution	18
Fig. 3.11 Comparison of three pilot structures [10].....	18
Fig. 3.12 Pilots location and the OFDM subcarriers scheme.....	20
Fig. 3.13 OFDM cyclic prefix	20
Fig. 3.14 Power spectrum of the samples	21
Fig. 3.15 Diagram of peak cancellation CFR technique [13]	22
Fig. 3.16 Distribution of generated signal amplitude.....	23
Fig. 3.17 Spectrum and Amplitude comparison on CFR signal	24
Fig. 3.18 CCDF & PDF of PAPR with or without clipping and filtering.....	24
Fig. 3.19 Constellation decode without synchronization and equalization	25
Fig. 3.20 STO Estimation using CP minimum difference method.....	27
Fig. 3.21 CP-based CFO estimation with simulation data	28
Fig. 3.22 Procedure for pilots extraction.....	28
Fig. 3.23 Channel estimation results	29
Fig. 3.24 Constellation and decoded image with pilot equalization	30
Fig. 3.25 General power amplifier scheme.....	30
Fig. 3.26 SNR vs BER in different modulation schemes [18]	31
Fig. 3.27 Power vs frequency of different transistor technologies [21]	33
Fig. 3.28 Direct learning approach DPD diagram	36
Fig. 3.29 Block diagram and photograph of the testbed	38
Fig. 3.30 Input power sweep simulation of the PA [28].....	39
Fig. 3.31 PA performance with different supply settings without DPD.....	40
Fig. 3.32 Parallel coordinates plot of the PA configuration search results	41
Fig. 3.33 DPD Result of the fifth optimal configuration	42
Fig. 4.1 Zync-7000 SoC block diagram [32]	45
Fig. 4.2 Implementation considerations of the proposed drone communication system.....	49
Fig. 4.3 Hardware and software development flow [33].....	49
Fig. 4.4 The logic fabric and its constituent elements [34]	50
Fig. 4.5 Simplified DSP48E1 slice functionality [35]	51
Fig. 4.6 Different quantization modes [40].....	54

Fig. 4.7 Different types of array partitioning [38].....	55
Fig. 4.8 Block diagram of the proposed LUT-based DPD implementation	59
Fig. 4.9 DPD with different LUT size settings	60
Fig. 4.10 Automatic HLS DPD code generation flow	61
Fig. 4.11 DPD Performance GMP vs Py-LUT vs HLS-LUT	62
Fig. 4.12 System-level block design	64
Fig. 4.13 Address assignment of the design.....	65
Fig. 4.14 Power analysis of the implementation	65

LIST OF TABLES

Table 1.1. Typical ISM transceiver ICs [1].....	1
Table 2.1 Three levels of flexibility to custom drone communication system	4
Table 2.2. Telemetry link scheme for drone [4]	5
Table 2.3. Video link scheme for drone [4]	6
Table 3.1. Gain and loss factors.....	9
Table 3.2. Kalman filter procedure of the problem.....	11
Table 3.3. Discrete time EKF.....	12
Table 3.4. Comparison of different OFDM protocols	19
Table 3.5. SNR for BER at 0.1% in AWGN channel [18].....	31
Table 3.6. Amplifier classes and properties.....	32
Table 3.7. The optimal configuration without DPD	41
Table 3.8. DPD results with optimal and fixed configurations.....	43
Table 4.1. Implementation considerations.....	46
Table 4.2. Key features of the Zedboard	52
Table 4.3. Overflow and quantization modes	54
Table 4.4. IFFT performance and resources of different architecture.....	57
Table 4.5. DPD performance comparison	62
Table 4.6. Synthesised DPD performance and resources usage	63

ACRONYMS

ARM	Advanced RISC Machine
APSoC	All-Programmable SoC
APU	Application Processor Unit
AWGN	Additive White Gaussian Noise
AXI	Advanced eXtensible Interface
BER	Bits Error Rate
CAN	Controller Area Network
CFO	Carrier Frequency Offset
CFR	Crest Factor Reduction
CLB	Configurable Logic Block
CP	Cyclic Prefix
CPU	Central Processing Unit
CPWL	Canonical Piecewise-Linear
CRC	Cyclic Redundancy Check
DAC	Digital-to-Analog Converter
DDR	Double Data Rate
DFT	Discrete Fourier transform
DPD	Digital Predistorsion
DSC	Drone Small Cells
DSP	Digital Signal Processing
DSSS	Direct Sequence Spread Spectrum
DVR	Decomposed Vector Rotation
EASA	European Union Aviation Safety Agency
EKF	Extended Kalman filter
FF	Flip Flop
FFT	Fast Fourit Transform
FHSS	Frequency-hopping spread spectrum
FIR	Finite Impulse Response
FPGA	Field-Programmable Gate Array
FSBL	First Stage Boot Loader
FSK	Frequency-Shift Keying
FSPL	Free-Space Path Loss
GaAs	Gallium Arsenide
GaN	Gallium Nitride
GCS	Ground Control Station
GMP	Generalized Memory Polynomial
GP	General-Purpose port
GPS	Global Positioning System
HLS	High-Level Synthesis
HP	High-Performance port
ICI	Inter-Carrier Interference
IDE	Integrated Development Environment
IDFT	Inverse Discrete Fourier transform
IFFT	Inverse Fast Fourit Transform
II	Initiation Interval

IP	Intellectual Property
ISI	Inter-Symbol Interference
ISM	Industrial, Scientific and Medical
LDMOS	Laterally Diffused MOSFET
LiDAR	Light Detection and Ranging
LNA	Low-Noise Amplifier
LO	Local Oscillator
LUT	Look-Up Table
LWIP	Lightweight IP
MP	Memory Polynomial
MPU	Microprocessing Unit
OFDM	Orthogonal frequency-division multiplexing
PA	Power Amplifier
PAPR	Peak to Average Power Ratio
PL	Programmable Logic
POSIX	Portable Operating System Interface
PS	Processing System
PSK	Phase-Shift Keying
QAM	Quadrature Amplitude Modulation
QoS	Quality of Service
QSPI	Queued Serial Peripheral Interface
RAM	Random Access Memory
RC	Radio control
RF	Radio Frequency
RSSI	Received Signal Strength Indicator
RTL	Register Transfer Level
RX	Receiver
SDK	Software Development Kit
SDR	Software Define Radio
SiC	Silicon Carbide
SNR	Signal to Noise Ratio
SoC	System on a Chip
SPI	Serial Peripheral Interface
STO	Symbol Time Offset
TX	Transmitter
UART	Universal Asynchronous Receiver/Transmitter
UAS	Unmanned Aircraft System
VGA	Variable Gain Amplifiers

CHAPTER 1. INTRODUCTION

Motivation

Drones are devices flying in the air, far away from the operator. The most common way to send and receive data/commands is by using the wireless communication system. Usually, when selecting the transceiver for drones, one will first consider the data rate and the distance. For example, to send control commands, one possibility is the use of general transceiver ICs or modules that operate under low-frequency ISM band. These transceivers are easy to use, they provide serial interfaces like SPI, I2C with a user manual to help set up their communication, with the advantages of low power consumption and long coverage range. However, these transceivers offer low data rate that might be enough for sending control data, like mission waypoints and inertial states, but not sufficient for large-volume data such as video and LiDAR data. Table 1.1 lists some typical transceivers that operate in the ISM band.

Table 1.1. Typical ISM transceiver ICs [1]

Name	Frequency	Modulation	Data Rate (Max)	Power Output	Sensitivity
CC2500	2.4 GHz	2FSK, GFSK, MSK, OOK	500 kbps	1 dBm	-104 dBm
CC1121	410 MHz ~ 480 MHz, 820 MHz ~ 960 MHz	2FSK, 2GFSK, 4FSK, 4GFSK, MSK, OOK	200 kbps	16 dBm	-120 dBm
NRF24L01	2.4 GHz	GFSK	2 Mbps	20 dBm	-94 dBm
SI4461	142 MHz ~ 1.05 GHz	4GFSK, GFSK, GMSK, OOK	1 Mbps	16 dBm	-129 dBm
ADF7021	80 MHz ~ 650 MHz, 862 MHz ~ 950 MHz	2FSK, 2GFSK, 3FSK, 4FSK, FHSS, GMSK, MSK	32.8 kbps	13 dBm	-130 dBm

In some drone applications, for example, drone small cells (DSC) [2], the drone is used as a base station. The communication system itself has the major role and it is supposed to transmit large volumes of data. For these special cases, the communication system needs to cover a large range, with the permission to transmit high power signal levels and occupy large bandwidths. The need to have a high-efficiency RF-frontend is of crucial interest, because the drone power supply system nowadays, apart from supplying the motors to maintain the drone position, few rooms remain to support a power-hungry wireless base station.

Furthermore, most drone communication systems are independent of the drone context. The communication system only involves the wireless data link and the flight controller only cares about how to fly. The communication system uses its maximum ability to have the best QoS (quality of service), conventionally sends signals with the same power, even though if the drone is fairly close to the GCS. Sending signals with a certain power level leads to unnecessary power consumption and is a kind of vulnerability that a third-party may utilize to estimate the location of the drone. Therefore, one of the objectives of this master thesis is to design a flexible and efficient communication system optimized for the UAS.

Objectives

Below listed the goals of this master thesis:

1. Provide an overview on the drone communication system.
 - Discuss current existing solutions on some popular drone platforms.
 - Identify the most used modulation, typical output power levels, bandwidth, data rates, power efficiency, coverage and operating frequencies.
2. Design a communication system architecture optimized for drones
 - Propose a scheme to adjust the TX power automatically.
 - Design an OFDM downlink communication system.
 - Include a DPD linearization system that in combination with the proposed TX power control scheme, significantly improves the overall power efficiency while maintaining the required linearity levels.
3. Implementation of core algorithms in a FPGA
 - Implement the main algorithms (IFFT and DPD) to IP cores for Xilinx FPGA by using the high-level synthesis (HLS) methodology to prove their feasibility.
 - Optimize the design by reducing the hardware resources usage and increasing the throughput to meet the required data rate.

Technologies involved

This thesis relies heavily on digital signal processing for RF communications. For example, IFFT is used in the OFDM signal generation. In order to have a better performance in linearity, DPD linearization is used to predistorted the baseband signal. In the scheme design part, Python programming language is used to build the mathematical models and perform the theory validation.

The FPGA programming is also part of the technologies involved in this thesis. To implement an IP core, techniques on in-chip interfaces, FPGA resources, and the hardware architectures have to be taken into account. The implemented IP core should give similar results as Python, so the cross-validation method is used to guarantee it. High-level-synthesis is used to develop and simulate an IP core. C++ is the selected language for the HLS development. Moreover, to reduce the validation time and avoid redundant coding in HLS, the template engine

technique is introduced. In short, this thesis widely covers both software development and FPGA hardware development.

Planning

A complete wireless communication system requires a huge amount of development for every specific detail and follows many industrial specifications. This thesis, however, will simplify many details and focus the new design strategies on the specific parts that can have an impact in the drone context. The project is therefore divided into three stages: design stage, implement stage and done stage. In time planning, general consideration will be done in advance to initiate some plan tasks as separate goals in the design stage. Then some tasks will be moved to the implement stage and focusing on these tasks, once a task is finished, it will be moved to the done stage. Each task can be considered independent and parallel, which means that there is no need to wait for a specific task. In this way, when encountering some difficulties, it is possible to switch to another task and maximize the final outputs within the project period. This is eventually following the Kanban scheduling system. The Kanban was designed to have three process stages: "design", "implement" and "done". Each task is a card in the Kanban board, initialized in the design stage and will be pushed to the next stage once been finished.

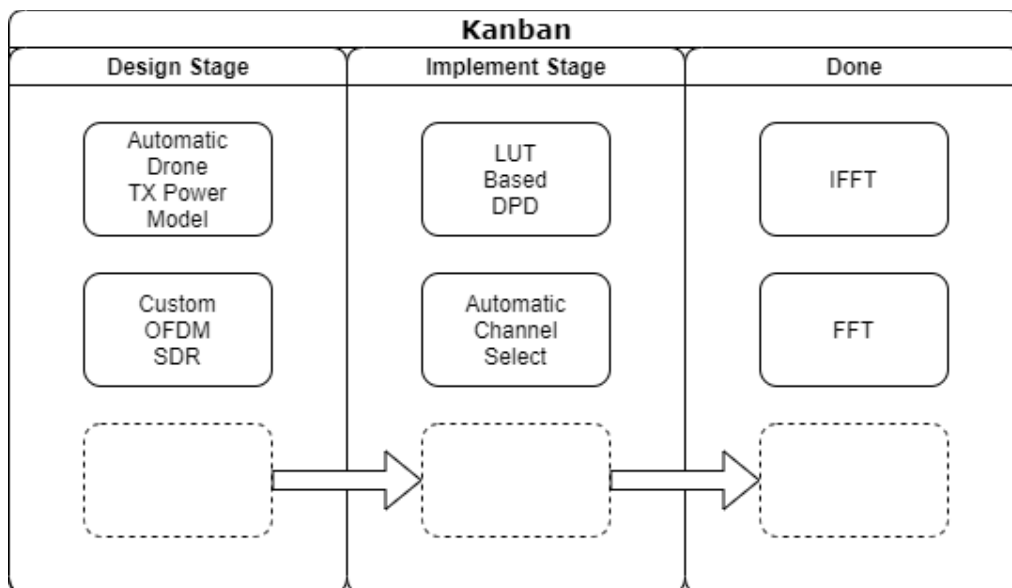


Fig. 1.1 Kanban board schedule

CHAPTER 2. DRONE COMMUNICATION SYSTEM

In this chapter, an overview on the drone communication system will be presented and discussed. Details on the transceiver type, modulation method, data rate, and power consumption of the existing commercial solutions will be analysed. According to these characteristics, we will analyse their design motivation, identify their advantages and try to minimize their disadvantages. Finally, we will come up with an optimized scheme for the drone communication system.

2.1. Targeting UAS

Before exploring the existing transmitting solutions, we will specify the UAS selected to apply the optimization on the communications system. Since drones are vehicles that fly at the altitude of tens and hundreds of meters, in order to perform data exchange, it is natural to think about using wireless solutions. UAS can be used for different purposes and have different levels of complexity. It can be as simple as a toy for kids or can be as complicated as a military weapon. For the simplest cases, the communication system is a trivial remote controller with only an uplink, so no data will return from the drone. Apparently, it is not worth spending efforts to perform optimization. According to the regulation from EASA [3], drones were classified into 5 classes, C0-C5. Each class follows with some limitations on the features such as maximum take-off mass (MOTM), height and speed. For example, C0 operates in the open category, only allows flying a low altitude (≤ 120 m) with MOTM of 250 g and air speed of 19 m/s. Although EASA does not explicitly regulate the communication system of UAS, some limitations on certain characteristics indirectly affects the communication system. For example, the maximum power voltage for C0 and C1 is 24 V, for C2 and C3 is 48 V. Since high-power power amplifiers require a supply voltage of 28 V, it is not possible to equip it on the C0 or C1 UAS. For those who can get permission to fly C4 drones, they are also more likely to gain special permission on the use of RF, to send higher power signals and to occupy a larger bandwidth. Table 2.1 shows the level of flexibility to perform modification on the drone communication system.

Table 2.1 Three levels of flexibility to custom drone communication system

	Low	Medium	High
Possible Class	C0,C1	C2,C3	C4
Data Rate	Low	High	Even Higher
Max Altitude	120 m	120 m	N/A
Weight	≤ 900 g	≤ 25 kg	N/A
Power Voltage	≤ 24 V	≤ 48 V	N/A
Worth to optimize	No	Yes	Yes

C2, C3 covers most of the cases, because it allows a drone weight up to 25kg, so it is possible to mount payloads such as LiDAR and cameras. With granted permission, it is possible to fly beyond the visual line of sight (BVLOS). In the view of communication system, it means both uplink and downlink are included. The transceivers must have enough bandwidth to transmit/receive video data, and must send them with enough power to cover a certain range. When the required TX power is getting higher, it makes more sense to give optimization on power efficiency. For the most special cases like C4 and sometimes the military drones that are granted with the highest permission, the flight altitude can be much higher and the TX power is much higher, there is more payload data to send and higher signal bandwidths can be used. The PA power efficiency versus linearity trade-off becomes more important and the proposed communications architecture would be much more useful.

2.2. Existing solutions

With considerations of robustness and range for telemetry, and the large throughput requirement for video transmission, most of the existing drones detach the communication affairs to the telemetry link and video link. According to the research of drone communication protocols in [4], popular modulation schemes have been proposed, as shown in the Table 2.2, spread spectrum techniques of frequency hopping spread spectrum (FHSS) and direct sequence spread spectrum (DSSS) are widely used in the telemetry link.

Table 2.2. Telemetry link scheme for drone [4]

Brand	Frequency	Modulation	Technology
DJI Phantom	2.4/5.8 GHz	FHSS/ DSSS	FASST/ Lightbridge
Futaba	2.4 GHz	FHSS/ DSSS	FASST
Spektrum	2.4 GHz	FHSS/ DSSS	DSM2/ DSMX
JR	2.4 GHz	FHSS/ DSSS	DMSS
Hitec	2.4 GHz	FHSS/ DSSS	AFHSS
Graupner	2.4 GHz	FHSS/ DSSS	HOTT
Yuneec	2.4 GHz	DSSS	ZigBee
Parrot AR2	2.4 GHz	OFDM	Wi-Fi
Immersion	433 MHz	FHSS	EZUHF

The telemetry link is used to send essential status information like battery voltage, position, and commands. These data are important for the operator but might not necessary for the application. On the other hand, these low-speed data need to be reliable and secure, hence spread spectrum techniques are extremely fit with them. According to the specification of DJI Phantom [5], when running in the RC and FCC mode (only enable the telemetry), without obstacles and free of interference, it is possible to fly at a distance of 7000 meters. It is enough for most drone applications.

Things are different when considering video streaming. The video stream contains much more data and requires more bandwidth than the telemetry link. Advanced modulation techniques such as OFDM is more likely to be used. The following table (Table 2.3) shows that FM and OFDM are popular choices.

Table 2.3. Video link scheme for drone [4]

Brand	Frequency	Modulation	Technology
DJI	2.4 GHz	OFDM	Lightbridge/Wi-Fi
Immersion	2.4 GHz	FM	-
Yuneec	5.8 GHz	OFDM	Wi-Fi
Connex	5.8 GHz	OFDM	-
Boscam	5.8 GHz	FM	-

The RF signal of the Video link occupies a large bandwidth, spread spectrum is no longer suitable because it will make the spectrum wider and difficult to meet the ISM regulations. The DJI phantoms radio link with the video stream enabled downgrades the maximum distance to 2000 meters. To find out more characteristics of the drone video link, a simple experiment was done with the DJI Lightbridge and a spectrum analyser. The Lightbridge can transmit both the video stream and the control commands. After the booting sequence, the signal from the Lightbridge can be monitored in the spectrum analyzer as shown in Fig. 2.1.

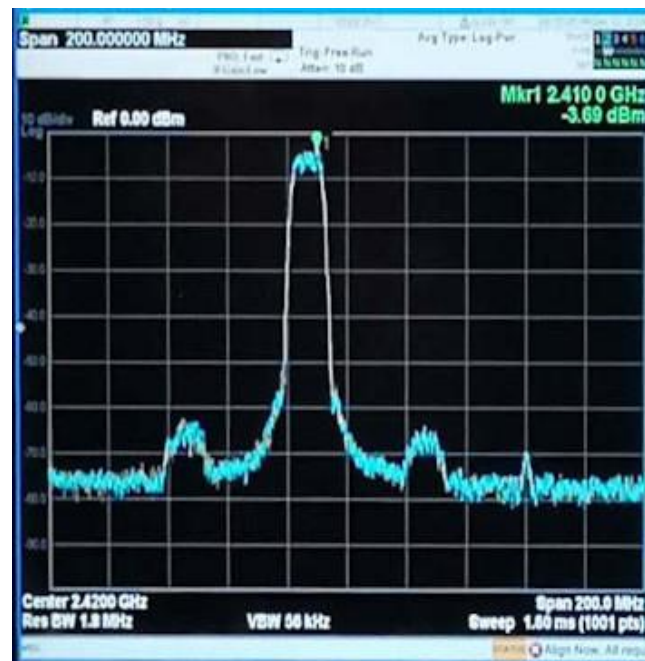


Fig. 2.1 Spectrum of the DJI Lightbridge signal

The spectrum analyzer received the signal through an antenna at a 2.4 GHz with a 10 dB attenuator. From the spectrum plot, the feature of the video signal can be clearly seen, it occupies 20 MHz in the 2.4 GHz band. If the drone position stays the same, the signal spectrum will always have the same shape and the

same power, as specified in Table 2.3, this is a 20 MHz OFDM signal. If the drone moves away from the receiver antenna, the signal power will decrease following a linear-like pattern. It means that the feature of the video link is possible to recognise and the fixed output power makes it possible for estimating the drone distance. Some drone detection projects [6, 7] were made based on these assumptions. Furthermore, the Lightbridge specs [8] shows that it transmits 100 mW at 2.4 GHz and the air system operating current is 650 mA at 12 V, so the overall system power efficiency for transmitting is very low, approximately $100\text{mW}/(650\text{mA}\cdot 12\text{V}) \cdot 100\% = 1.28\%$. (It is not the PA efficiency alone, the PA efficiency must be higher, but we do not have the detailed information to know the specific PA efficiency)

The camera is a type of payload equipped on the drone, so the video link can also be considered as the payload link responsible for transferring payload data in real-time. So far, the conclusion on the conventional video link is listed below:

- Video (payload) link uses OFDM for sending a large amount of data with high spectral efficiency (i.e., bits/s/Hz).
- The output power is fixed regardless of the drone position.
- The wireless traffic is asymmetric and it is more relevant in the downlink.
- The RF system efficiency (defined as the RF transmit power/consumed DC power x100%) is low.

Inspired from the above conclusions, the following subchapter will give a scheme to optimize the communication system for UAS.

2.3. Optimized transmitter architecture

The proposed architecture is an OFDM communication system optimized for drone communications. All the modifications were performed in the drone transmitter side. Fig. 2.2 shows the block diagram of the proposed scheme.

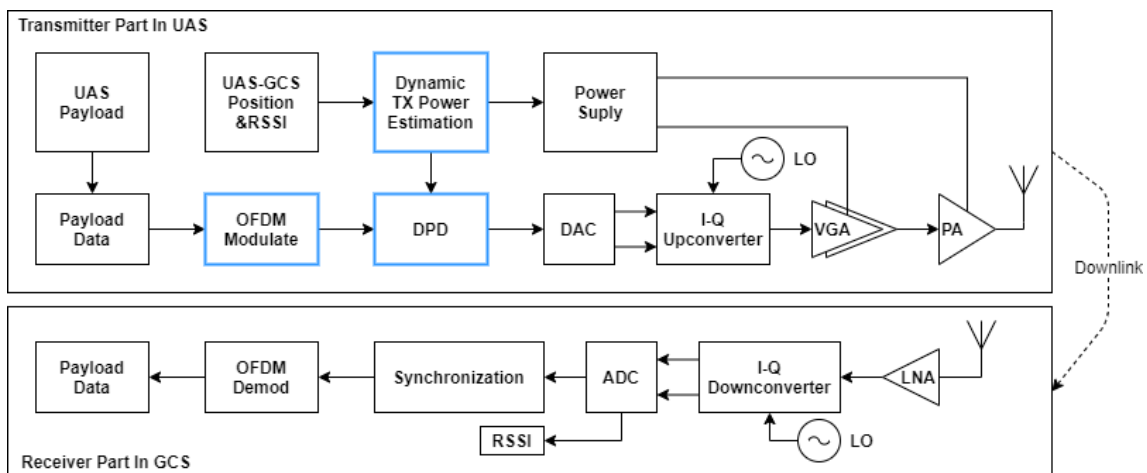


Fig. 2.2 Block diagram of the drone transmitter and GCS receiver.

Fig. 2.2 shows the three subsystems (highlighted in blue) specifically designed in this thesis, which took the most of our efforts and resources. First, because of the wide use of OFDM in drone video links, it deserves to remain in the scheme. Second, the dynamic TX power estimation block estimates the required TX power, based on the distance between UAS and GCS, and the RSSI from the GCS receiver. With the estimated TX power level, the power supply can adjust the gain of the VGA and the supply voltage of the power amplifier in order to improve the PA power efficiency. Finally, the DPD is used to guarantee the PA linearity. The change of the PA supply voltage will lead to a change of the PA behaviour. So, the DPD needs to be able to track these changes and adapt its coefficients.

The automatic TX power control system can add some advantages to the drone communication system. For example, if multiple drones are flying at the same time and operating at the same radio channel, each user will have to keep a distance to avoid interference. The power control system can automatically lower the TX power and make it possible to fly more drones within an area. On the other hand, a fixed output power makes it possible for a third party to detect the existence of drone and locate the position. The dynamic TX power can at least hide the drone distance information. When the drone is closer to the operator than the RF detector, the drone detector has more difficulties to detect the drone's existence.

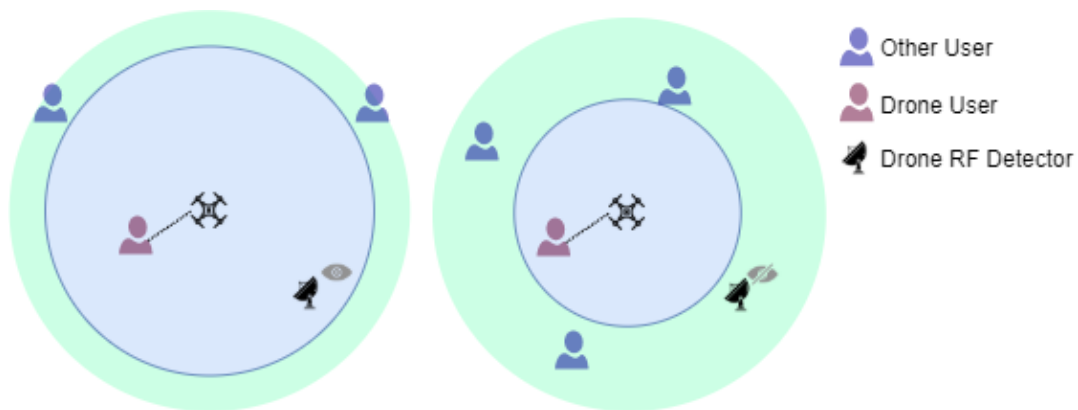


Fig. 2.3 Static TX power vs dynamic TX power.

In summary, the proposed scheme can improve the UAS downlink with the following advantages:

- It can save the unnecessary TX power.
- It makes the drone hard to be detected by a third party.
- More drones could be used within a large area.
- It can improve PA efficiency by using optimized supply settings and DPD.

CHAPTER 3. SYSTEM DESIGN

This chapter will present the design details of the communication system. First, for estimating the required TX power, a Kalman filter will be designed. Then the OFDM signal generation and recovery will be designed and simulate. Finally, the DPD algorithms will be presented and tested together with the OFDM signal in a customized testbed built for the occasion.

3.1. Automatic transmit power control

Link budget is a general method to calculate the required TX power. In the link budget calculation, the received power is estimated by adding all gains and subtracting all losses. Note that the unit of power in this thesis is always expressed in the logarithmic scale (i.e., dBW or dBm). The equation of the link budget is shown below.

$$P_{rx} = P_{tx} + \sum_i Gain_i - \sum_j Loss_j \quad (3,1)$$

In the proposed drone communication system, the gain items include the TX and RX antennas gain, and the gains from RX LNA and the TX PA. The loss items are the free space propagation loss (FSPL) and the pointing loss due to the moving behaviour, and the antenna polarization. Some of these items are constant, for example, the RX LNA gain is always fixed, while other items change according to the situation, for example, the free space loss changes with the distance. Table 3.1 shows detail of the loss and gain factors taken into account in this master thesis.

Table 3.1. Gain and loss factors

Factors	Fixed
GCS antenna gain	Yes
GCS LNA gain	Yes
GCS cabling loss	Yes
GCS pointing loss	No
UAS antenna gain	Yes
UAS cabling loss	Yes
UAS pointing loss	No
UAS PA gain	No
Free space propagation loss	No

Although the UAS pointing angle can be obtained from the IMU data, the GCS can be a hand-held device whose angular position is unpredictable, hence the pointing loss cannot be calculated. For simplicity, assuming the use of bipolar antennas, a constant pointing loss budget of 6 dB (the worst case) is reserved. As a result, the remaining variables are the TX PA gain and the FSPL. Denote P_c

as the sum of all the constant losses and gains, the received power P_{rx} can be derived as follows.

$$P_c = P_{tx} + \sum G_{const} - \sum L_{const} \quad (3,2)$$

$$P_{rx} = P_c + G_{PA} - L_{freespace} \quad (3,3)$$

The objective is to estimate the required TX power and then derive the PA gain. The OFDM system sensitivity and the receiver performance decide the required received power, which is also a constant. In the actual practices, an extra link margin will be preserved to the system. By moving the PA gain to left side, the equation of the required PA gain is shown below, where P_{cc} denotes the sum of all constants.

$$P_{cc} = P_{rx,required} - P_c + ExtraLinkMargin \quad (3,4)$$

$$G_{PA,required} = P_{cc} + L_{freespace} \quad (3,5)$$

P_{cc} can be obtained by some rough experimental tests. Now that the only variable left is the FSPL, which is the loss between two isotropic radiators in the free space, expressed as a power ratio. In the log scale, it is a function of distance and center frequency. The equation of FSPL is shown below, the unit of distance is meter and the unit of frequency is Megahertz.

$$FSPL(dB) = 20\log_{10}(d) + 20\log_{10}(f) - 27.55 \quad (3,6)$$

In an ideal case, given the accurate GPS position of GCS and the UAS, the distance can be calculated and the required TX power can be indicated. The position of GCS can be sent to the UAS through the telemetry link, and the automatic power regulation takes place on the UAS. However, in a realistic case we also have noise and unwanted obstacles, and thus this simple method is not reliable. The following subchapter describes the efforts of using a Kalman filter to make it more reliable and robust.

3.1.1. Kalman filter for estimating the RX power

There are several reasons for not having an accurate estimation. For example, the constant P_{cc} is a rough approximation, and the GPS location can have some meters of error. Apart from these numerical errors, obstacles such as buildings, mountains, and trees can randomly appear in the path of the wireless communication link. If the link margin is small, the UAS may lose communication for a moment. On the contrary, if a big link margin is introduced to the transmitter, then the TX power is significantly higher than the minimum required power, which is clearly against our objective of saving power. What's more, the GCS can also be a moving object such as a car, and its GPS location may be missing, so the distance measurement is not always available. Fortunately, the receiver on the

GCS can calculate the RSSI and send it to the UAS to improve the required TX power estimation.

The problem now becomes: Knowing the previous received signal power, predict the next RX power. The derived equation is shown below, where \hat{P}_{rx} represents the estimated RX power Δ represents the difference.

$$\hat{P}_{rx} = P_{rx} + \Delta G_{PA} - \Delta L_{freespace} \quad (3,7)$$

The difference between TX power (ΔG_{PA}) is a variable which can be controlled by the automatic power control system. The difference of the FSPL ($\Delta L_{freespace}$) can be derived from the distance. The current distance is denoted as d_t , and the former distance as d_{t-1} , the difference of FSPL can be derived as follows,

$$\Delta L_{freespace} = L(d_t) - L(d_{t-1}) \quad (3,8)$$

$$\Delta L_{freespace} = 20 \log_{10} (d_t) - 20 \log_{10} (d_{t-1}) \quad (3,9)$$

To be more specific, the distance can be measured by calculating the Euclidean distance from the UAS to the GCS by using the GPS data. By denoting the longitude as X , the latitude as Y and the height as Z (all the units are in meters), the distance, d , is given by

$$d = \sqrt{(X_{gcs} - X_{uas})^2 + (Y_{gcs} - Y_{uas})^2 + (Z_{gcs} - Z_{uas})^2} \quad (3,10)$$

The linear velocity v along the line of GCS and UAS can also be estimated from the distance difference over time. Finally, the prediction of RX power is a function over time, given by

$$\hat{P} = P + 20 \log_{10} \left(\frac{d + v \Delta t}{d} \right) + \Delta G_{PA} \quad (3,11)$$

In short, the system keeps tracking the RX power P , distance d and velocity v between GCS and UAS. The next state can be predicted by equation (3,11), and the measurement of distance and RX power can be obtained from the GCS through the telemetry link to improve the estimation. It can be described as a typical Kalman filter process as shown in the Table 3.2.

Table 3.2. Kalman filter procedure of the problem

State	p, d, v
Predict	$\hat{p}, \hat{d}, \hat{v} = f(p, d, v, \Delta t)$
Measurement	$gps \rightarrow d, rssi \rightarrow p$

The basic Kalman filter requires linear equations for state propagation, however, the logarithmic operation is not linear. Fortunately, the speed between GCS and UAS is not very high, extended Kalman filter can be used to linearize the problem.

As shown in Fig. 3.1, the relationship between FSPL and distance at 2.45 GHz, from 10 to 2000 meters is a log curve, but when zoom in and have a closer look of 100 meters range from 240 to 340 meters, it is nearly a straight line.

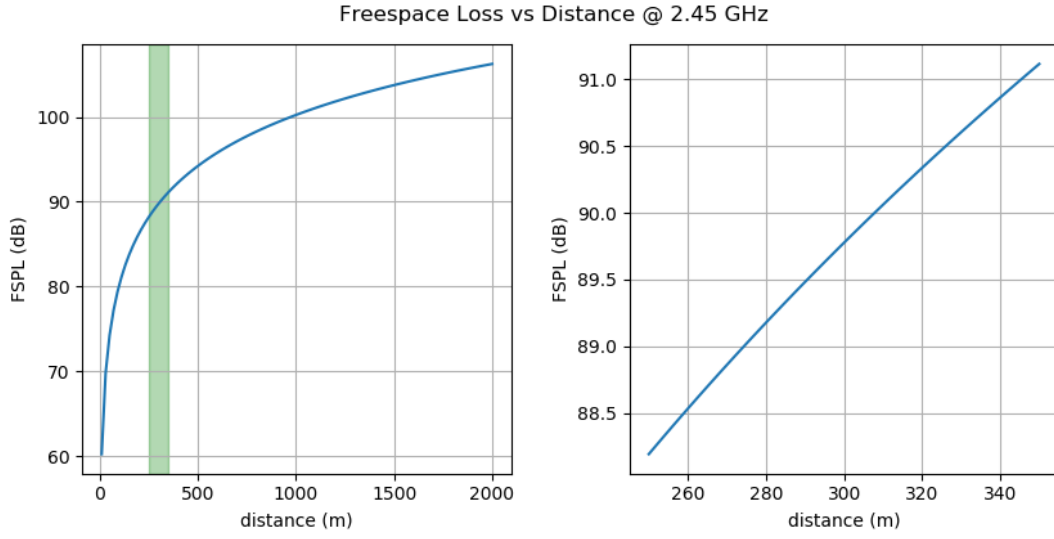


Fig. 3.1 FSPL vs distance

The discrete-time Extended Kalman filter process is shown in Table 3.3.

Table 3.3. Discrete time EKF

Predict	
State estimate	$\hat{x}_{k k-1} = f(\hat{x}_{k-1 k-1}, u_k)$
Covariance estimate	$P_{k k-1} = F_k P_{k-1 k-1} F_k^T + Q_k$
Update	
Measurement residual	$\tilde{y}_k = z_k - h(\hat{x}_{k,k-1})$
Residual covariance	$S_k = H_k P_{k,k-1} H_k^T + R_k$
Kalman gain	$K_k = P_{k k-1} H_k^T S_k^{-1}$
Updated state estimate	$\hat{x}_{k k} = \hat{x}_{k k-1} + K_k \tilde{y}_k$
Updated covariance estimate	$P_{k k} = (I - K_k H_k) P_{k k-1}$

The state prediction function f contains three individual functions for this model, one for predicting the RX power, one for the distance and one for the velocity.

$$\begin{bmatrix} \hat{p} \\ \hat{d} \\ \hat{v} \end{bmatrix} = f \left(\begin{bmatrix} p \\ d \\ v \end{bmatrix} \right) = \begin{bmatrix} p + 20 \log_{10} \left(\frac{d + v \Delta t}{d} \right) + \Delta G_{PA} \\ d + v \Delta t \\ v \end{bmatrix} \quad (3,12)$$

The F in the Extended Kalman filter is the Jacobians of f , given as

$$\mathbf{F} = \frac{\partial f}{\partial x} = \begin{bmatrix} \frac{\partial f_1}{\partial p} & \frac{\partial f_1}{\partial d} & \frac{\partial f_1}{\partial v} \\ \frac{\partial f_2}{\partial p} & \frac{\partial f_2}{\partial d} & \frac{\partial f_2}{\partial v} \\ \frac{\partial f_3}{\partial p} & \frac{\partial f_3}{\partial d} & \frac{\partial f_3}{\partial v} \end{bmatrix} \quad (3,13)$$

$$= \begin{bmatrix} 1 & \frac{20d(\frac{1}{d} - \frac{1}{d^2}(d + \Delta tv))}{(d + \Delta tv)\ln(10)} & \frac{20\Delta t}{(d + \Delta tv)\ln(10)} \\ 0 & 1 & \Delta t \\ 0 & 0 & 1 \end{bmatrix}$$

Similarly, to calculate the Jacobians of the measurement function h , the measurement matrix \mathbf{H} is given by

$$h \left(\begin{bmatrix} p \\ d \\ v \end{bmatrix} \right) = \begin{bmatrix} p \\ d \end{bmatrix}, \mathbf{H} = \frac{\partial h}{\partial x} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \quad (3,14)$$

The EKF also requires the process noise matrix \mathbf{Q} and the measurement noise matrix \mathbf{R} . The processing noise of RX power is independent of the position, and the processing noise of velocity and distance can use the discrete white noise model in [9], \mathbf{Q} is given below and σ_{rx} represents the RX power variance and σ_d is the distance variance.

$$\mathbf{Q} = \begin{bmatrix} \sigma_{rx}^2/\sigma_d^2 & 0 & 0 \\ 0 & T^4/4 & T^3/2 \\ 0 & T^3/2 & T^2 \end{bmatrix} \sigma_d^2 \quad (3,15)$$

For the measurement noise, the noise from the receiver of GCS is independent of the GPS. The measurement noise matrix \mathbf{R} can be defined by doing some experimental measurements.

$$\mathbf{R} = \begin{bmatrix} \sigma_{receiver}^2 & 0 \\ 0 & \sigma_{gps}^2 \end{bmatrix} \quad (3,16)$$

The proposed EKF design has been coded in Python, and numerical simulation was conducted to validate the correctness of the theory. The simulation results are presented in the following subchapter.

3.1.2. Simulation of the power and distance estimation in Python

The first simulation assumed a UAS flying away from the GCS without any obstacle, the distance and RSSI data were generated according to the mathematical model as the “true value”. Gaussian noise was added to the “true value” to emulate the sensor measurement. The proposed EKF filter was used to predict and update the states. Fig. 3.2 shows the drone moving away with a

speed of 10 m/s, the EKF can work fine to follow the true line of RX power. The distance in the following simulation are the Euclidean distance between the drone and the GCS.

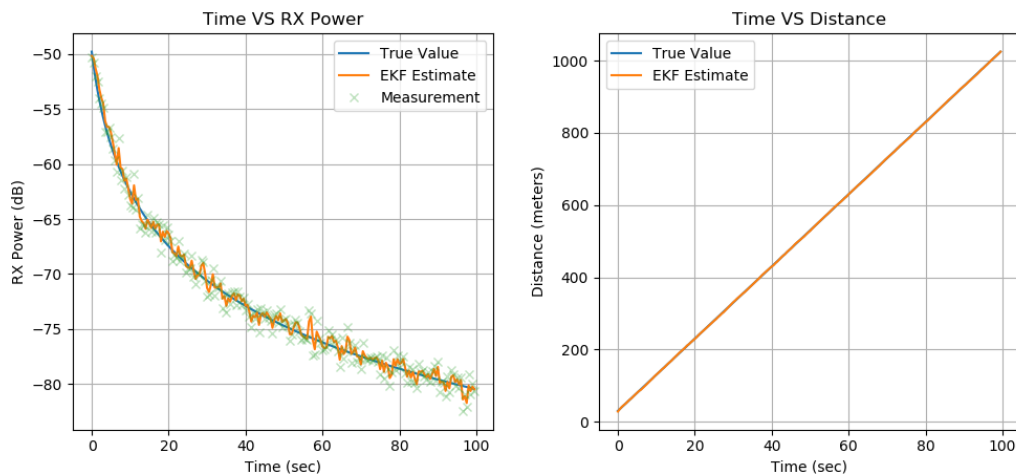


Fig. 3.2 EKF RX power and distance vs time

The second simulation in Fig. 3.3 takes the obstacles into account. Let's assume that the UAS signal is blocked by some object (e.g., buildings, trees, mountains, etc.) at 400 meters and 600 meters, and each obstacle can cause 6 dB of loss. The same EKF was applied to the emulated data, as shown in Fig. 3.3, the filter can match the line of true value, which validates the robustness of the Kalman filter for predicting the RX power and distance.

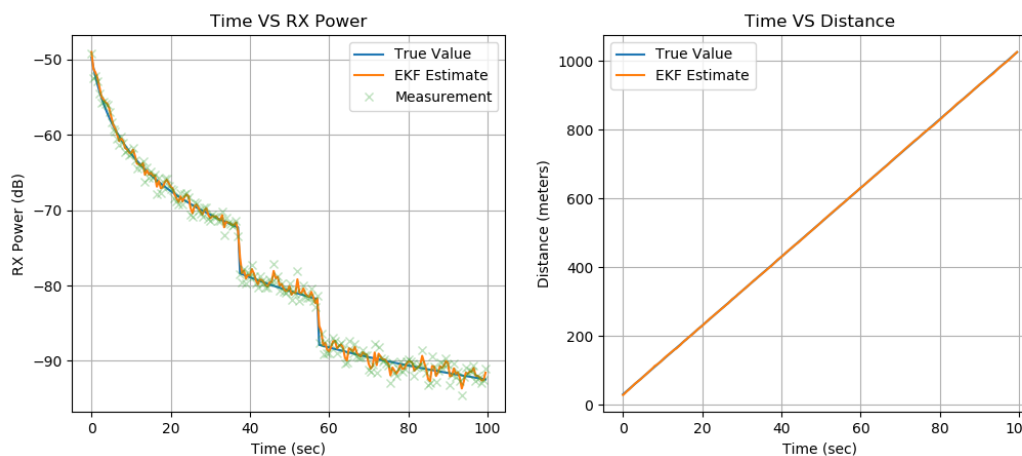


Fig. 3.3 EKF RX power and distance vs time with obstacles

A last simulation has been considered to complete some of the typical scenarios. Let's assume that the UAS will hold the position for 10 seconds (maybe executing some missions) and then fly back to GCS. As shown in Fig. 3.4, the change of velocity does not affect the quality of the EKF estimation.

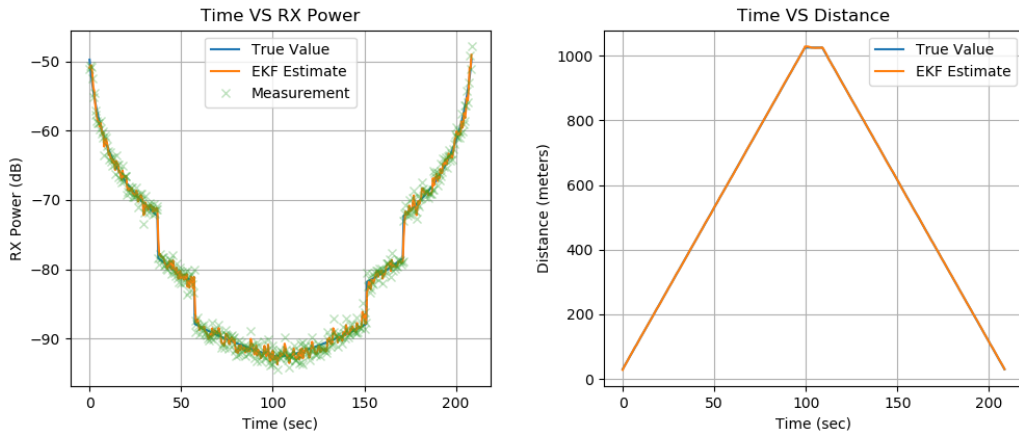


Fig. 3.4 EKF RX power and distance vs time with obstacle and go back

3.1.3. TX power control logic

Taking into account the good estimation of the RX power obtained with the EKF, it is possible to calculate the required PA gain by using equation (3,5) and adjust the TX power. In this work, we used a simple strategy of increasing/decreasing the TX power when the prediction exceeds a certain margin. For example, given the margin of 6 dB and the required RX power of -70 dBm, when the estimated RX power is lower than -73 dBm or higher than -67 dBm, the output power will increase or decrease 3 dB. To adjust the output power, the VGA gain and the power supply of PA need to be set. In the meantime, the DPD linearizer needs to be informed and adapt its coefficients. With this method (summarized in Fig. 3.5), it is possible to pre-define a set of output power levels, which is helpful for the DPD to have a limited number of pre-calculated coefficient configurations.

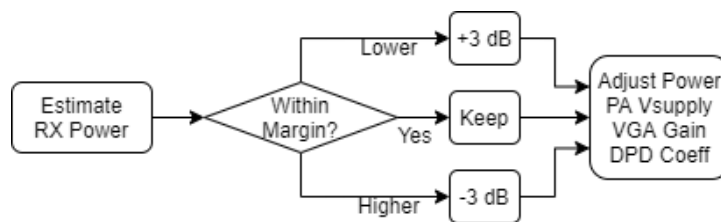


Fig. 3.5 Automatic power control logic

In combination with the third simulation from the last subchapter, let's assume that the UAS flew away from the GCS and then waited 10 seconds and came back. The behaviour of the automatic TX power control was simulated and the results are shown in Fig. 3.6. In Fig. 3.6-left, the orange line shows that the RX power was well controlled at the level of -70 dBm. The red line shows the TX PA gain increase and decrease with a similar shape to the distance plot.

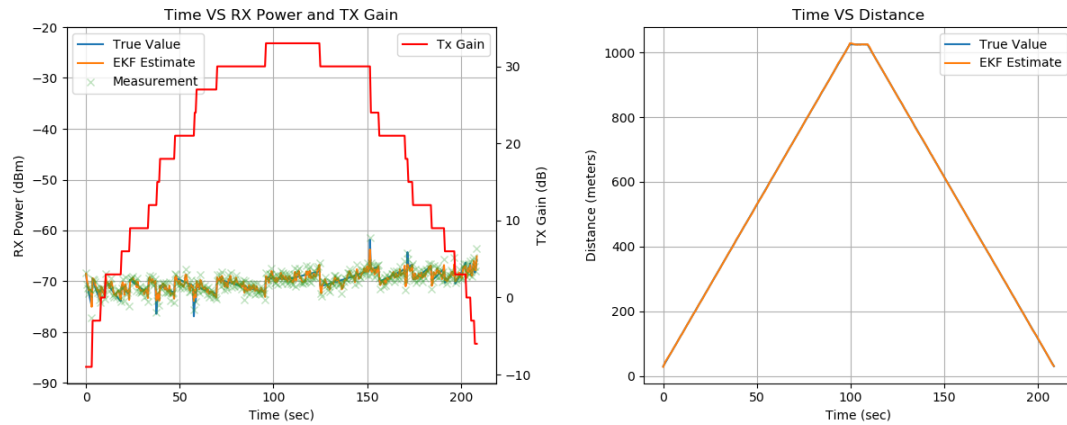


Fig. 3.6 The simulated control of TX power

With the proposed automatic TX power control logic, the TX power can change according to the specific needs and thus saving unnecessary power consumption. To improve the PA efficiency while guaranteeing its linearity, DPD linearization will be included, it will be described later in subchapter 3.3. Before that, the OFDM system needs to be designed for signal generation. Next subchapter will discuss the design of a simple OFDM downlink system.

3.2. Design of an OFDM system

Orthogonal frequency division multiplexing (OFDM) is a suitable multicarrier access technique for the drone video link, since it allows achieving high data throughput in the downlink. As shown in Fig. 3.7, full implementation of an OFDM system involves the design of a bit interleaver, mapper, synchronizer, etc., which is not the objective of this thesis. However, generating a decent OFDM signal is necessary for demonstrating the application. In this work, the OFDM was simplified, most of the efforts were put in the signal generation.

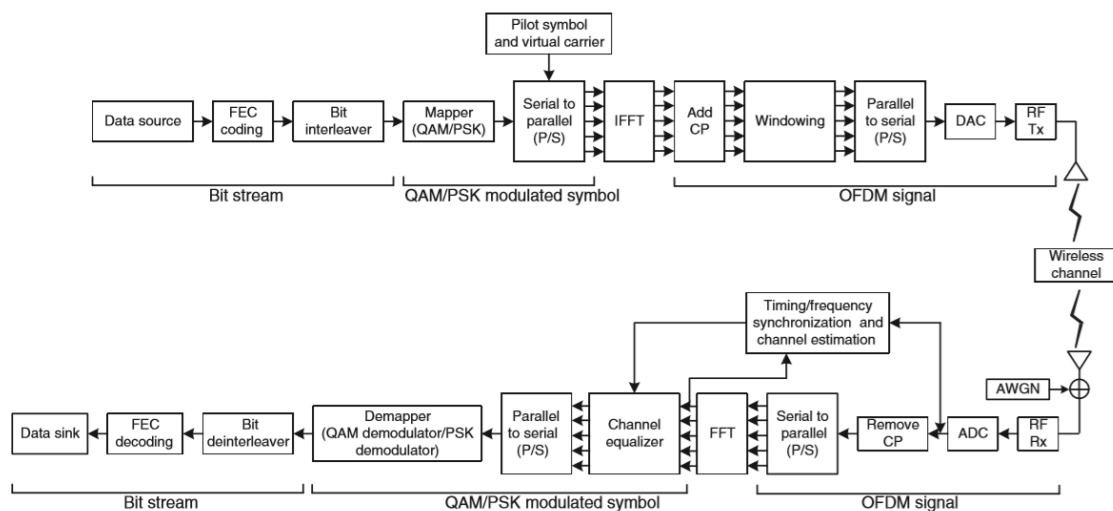


Fig. 3.7 OFDM full implementation diagram [10]

As the UAS downlink usually contains a video stream, an image was selected as representative of the payload data. The image data bytes will go through the encoder, QAM mapper and other function blocks to generate the baseband I-Q signals. The baseband data will then be upconverted to the RF signal and transmitted through the air. The block diagram in Fig. 3.8 shows the simplified OFDM generator used in this thesis.



Fig. 3.8 OFDM baseband generator

3.2.1. Encoder

The encoder converts the RGB image bytes to bits array follows the order of big-endian. To lower the design complexity, compression methods are not used, so the length of bits has a direct relationship with the size of the image, and the frame rate is low. Besides, retransmissions and CRC methods are not included. The received image could have lots of distortion depending on the transmission quality.

3.2.2. QAM mapper

Typically, the OFDM transmitter requires the bits mapped into PSK or QAM symbols which will be subsequently converted into N parallel streams. The modulation scheme 64-QAM was used in this work, with a square-like constellation and Gray coding as shown in Fig. 3.9.

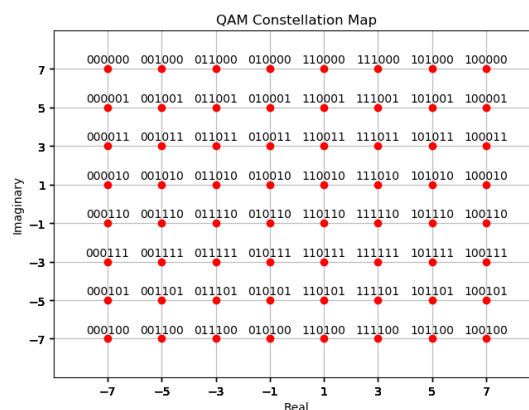


Fig. 3.9 QAM-64 gray coding constellation

Each 64-QAM symbol contains six bits of information, so the number of payload data in bits should be an integer multiple of six. As the encoder directly converts the 3-channel (RGB) image from bytes to bits, it is possible to group the data into 6 bits per batch. The equations below show the relationship between RGB image

size and the number of bits and symbols, where w and h denote the image width and height respectively, and n denotes number of symbols per image.

$$\text{ImageBits} = w \times h \times 3(\text{rgb}) \times 8(\text{bits/byte}) = n \times 6 \quad (3,17)$$

$$\text{SymbolsPerImage} = n = w \times h \times 4 \quad (3,18)$$

In short, the QAM mapper converts every 6 bits to a complex symbol according to the constellation map in Fig. 3.9 and then passes it to the next procedure. The QAM mapping was done in Python and Fig. 3.10 shows the symbol distribution of the “lenna.jpg”. The heat map on the right shows the distribution of symbols in percentage. All QAM symbols are used with the minimal distribution of 0.4% and the maximal of 3.1%, over 1,048,576 symbols.

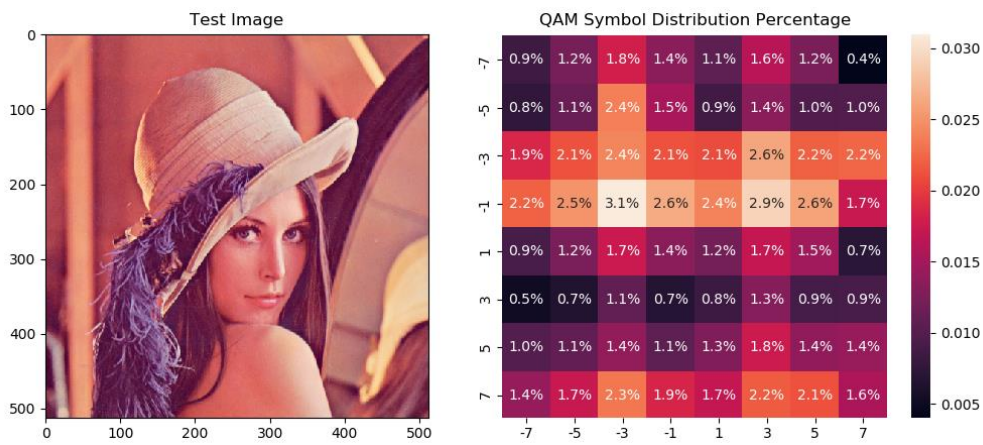


Fig. 3.10 Sample image and QAM symbol distribution

3.2.3. Pilots and serial to parallel conversion

OFDM time-domain signals consist of a set of data subcarriers that deliver the information and pilot subcarriers that help to estimate the radio channel. In the frequency-domain, data carriers are the QAM symbols, which have been generated in the previous subchapter by the mapper. Pilot carriers are inserted between data carriers or OFDM symbols depending on the arrangement. Three different types of pilot structures are considered: block type, comb type, and lattice type. Fig. 3.11 shows the three aforementioned arrangements.

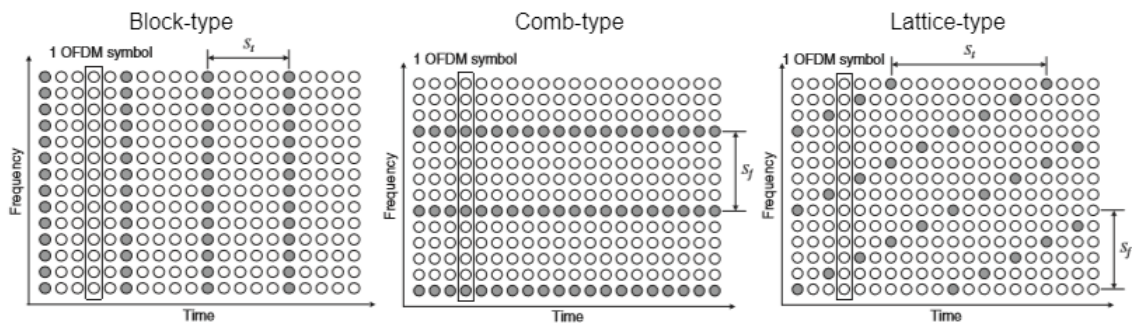


Fig. 3.11 Comparison of three pilot structures [10]

In the block-type, the OFDM symbols with pilots at all subcarriers are transmitted periodically. Time-domain interpolation is performed to estimate the channel along the time axis, it is suitable for frequency-selective channels. For the Comb-type, every OFDM symbol has pilot tones that are periodically-located between subcarriers. Frequency-domain interpolation is performed to estimate the channel along the frequency axis, it is suitable for fast-fading channels. In Lattice-type, pilot tones are inserted along both the time and the frequency axes with certain periodicity. Both time and frequency domain interpolations are required to estimate the channel, it can work for both frequency-selective and fast-fading channels, however, it is complicated to implement. According to the study of the UAS communication channel [11, 12], for the air to ground communication during the take-off, en-route and landing phases is more justified to use the fast-fading channel, hence, the comb-type has been selected in this thesis.

Each OFDM symbol contains several QAM symbols, these symbols have to be parallelized before sending them to the IFFT block. The number of QAM symbols per spectrum shot depends on the number of OFDM subcarriers and the number of pilot subcarriers. For example, in the WLAN 802.11n/ac protocol, 20 MHz bandwidth is divided by 64 for each subcarrier, the space between each subcarrier is 312.5 KHz. Only 56 subcarriers near the low frequency are used with 52 subcarriers for data and 4 for pilots. In LTE, typically 2048 subcarriers are used, but only 1200 subcarriers are active, with subcarrier spacing of 15 kHz and sampling frequency of 30.72 MHz, the occupied bandwidth is approximately 18 MHz. The following table shows a comparison of different standards using OFDM-based communications.

Table 3.4. Comparison of different OFDM protocols

	LTE (BW=20MHz)	WIFI 802.11g/a	WIFI 802.11n/ac
Total Subcarriers (N _{IFFT})	2048	64	64
Active Subcarriers	1200	52	56
Subcarrier spacing Δf	15 kHz	312.5 kHz	312.5 kHz
Sampling frequency (F _s)	30.72 MHz	20 MHz	20 MHz
Occupied Bandwidth	18 MHz	16.25 MHz	17.5 MHz

In this work, we considered the 802.11n/ac as the OFDM design reference. So, 64 IDFT with 56 activated subcarriers are used. The pilot and subcarriers allocation are shown in Fig. 3.12.

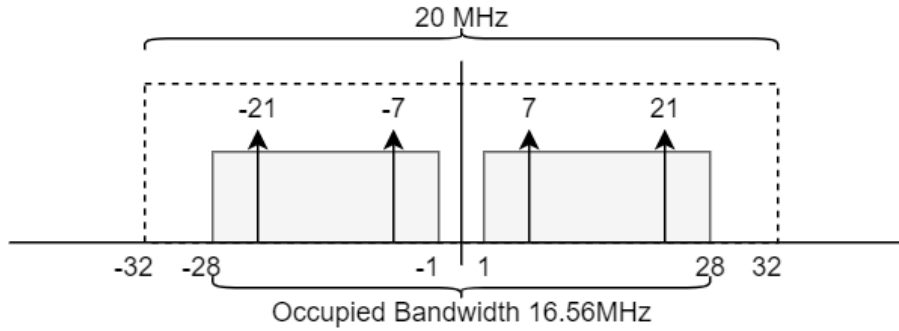


Fig. 3.12 Pilots location and the OFDM subcarriers scheme

The serial to parallel conversion requires grouping every 52 QAM symbols to a batch. However, the number of QAM symbols of an image cannot fit with the size of integer multiple of 52, redundant random symbols were added at the end to address the lack of OFDM symbols.

3.2.4. IFFT

Time-domain OFDM baseband signals were given by applying the IDFT (Inverse Discrete Fourier Transform) of the frequency-domain OFDM symbols. It can be computed efficiently by using the IFFT (Inverse Fast Fourier Transform) algorithm.

$$x_l[n] = \frac{1}{N} \sum_{k=0}^{N-1} X_l[k] e^{j2\pi kn/N} \quad (3,19)$$

With the IFFT functions from the Python library of *NumPy*, this procedure can be easily done with one line of code. IFFT should apply to every OFDM symbol individually and ensure that the same IFFT size was used.

3.2.5. Cyclic prefix

The OFDM needs guard intervals to mitigate the ISI (Inter Symbol Interference). The length of the guard interval should be set not shorter than the maximum delay (i.e., delay spread) of the channel. Cyclic prefix (CP) consists in extending the OFDM symbol by copying the last samples of the time-domain symbol at the beginning. Taking a quarter of the 64 IFFT size signal as the CP, the last 16 samples will be copied at the beginning, as shown in Fig. 3.13.

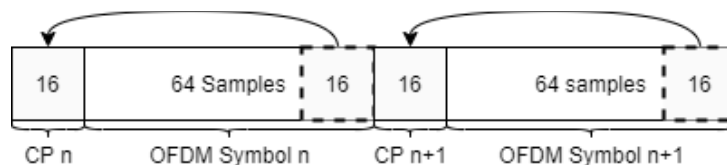


Fig. 3.13 OFDM cyclic prefix

3.2.6. OFDM baseband data samples

After adding the CP, the original 512x512 RGB image has been converted to baseband data with 1,613,200 samples. The samples are then divided into 10 batches (this would not break an OFDM symbol); each batch has 161,320 samples that can be directly sent to the I-Q upconverter after digital to analog conversion. In the following chapter, we will use these data samples for training the DPD to guarantee the required linearity levels at the output of the RF power amplifier and enhance its power efficiency. To better describe the generated data, Fig. 3.14 shows the power spectrum of the I-Q baseband signals, samples were normalized to 1 before the FFT.

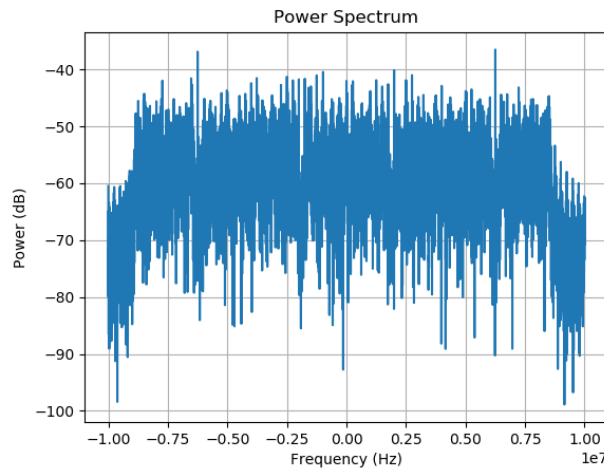


Fig. 3.14 Power spectrum of the samples

One unwanted characteristic of OFDM signal is that it can present high peak to average power ratios (PAPRs). The transmit signals can have high peak values since many subcarrier components can be added constructively (in-phase) after the IFFT operation. The calculation of PAPR is given as below.

$$PAPR(dB) = 10 \log \left(\frac{P_{max}}{P_{avg}} \right) = 10 \log \left(\frac{\max(|x[n]|^2)}{\text{avg}(|x[n]|^2)} \right) \quad (3,20)$$

The PAPR can degrade a wireless transmission chain since it can put most of those systems operating in a large-signal nonlinear zone. The DAC and the PA of the transmitter require large dynamic ranges to avoid amplitude clipping. Crest factor reduction (CFR) techniques are therefore used to reduce the PAPR of the OFDM-based signals. Several CFR techniques for OFDM-based signals have been published in literature, for example:

- **Coding**, that consists in using custom coding schemes to reduce the occurrence probability of the same phase of signals by selecting the codewords that minimize the PAPR in the transmission.
- **Partial Transmit Sequence (PTS)**, that partition the input data into disjoint sub-blocks. Then, the sub-carriers in each sub-block are IFFT transformed into time-domain partial sequences and independently weighted by phase factors.

- **Selected Mapping (SLM)**, that use the transmit sequence and multiply it by a codeword (vector of phase-shifts) that will change it in order to decrease the PAPR.
- **Interleaving Technique**, similar to the SLM, uses a set of interleavers instead of phase sequences to generate new data blocks to reduce PAPR of OFDM-based signals.
- **Clipping and Filtering**, that is used for abruptly reduce the peaks on the envelope signal. Nonlinear distortion of spectral regrowth will occur due to the peak cut-off, following a band-pass filter is used to compensate for the out-of-band distortion, however residual in-band distortion remains.

As explained in [13], the peak cancellation (PC) technique can reduce the signal PAPR and properly combined with DPD can enhance the PA power efficiency while guaranteeing the overall linearity. The PC method belongs to the clipping and filtering branch of CFR techniques. The block diagram of scaled peak cancellation (SPC) technique is shown in the figure below.

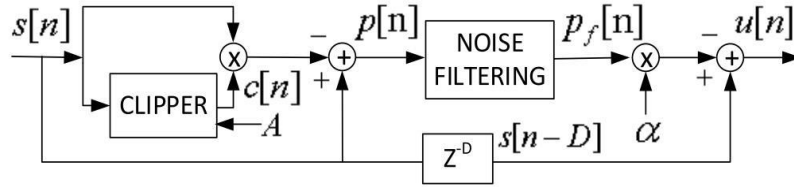


Fig. 3.15 Diagram of peak cancellation CFR technique [13]

The clipper output $c[n]$ can be written as (3,21) where A is the clipping threshold.

$$c[n] = \begin{cases} A/|s[n]| & \text{if } |x[n]| > A \\ 1 & \text{if } |x[n]| \leq A \end{cases} \quad (3,21)$$

The clipped pulse $p[n]$ is written as follows.

$$p[n] = s[n] - s[n] \cdot c[n] \quad (3,22)$$

After applying low pass filter to the clipped pulse to remove the high frequency discontinuity, and scaling by a factor α , the final PAPR reduced signal $u[n]$ can be obtain by subtracting a delay version of the input $s[n]$, the output $u[n]$ is described as follows, where $h[n]$ is the low pass filter.

$$\begin{aligned} p_f[n] &= p[n] * h[n] \\ u[n] &= s[n - d] - \alpha \cdot p_f[n] \end{aligned} \quad (3,23)$$

Scaled repeated peak cancellation (SRPC) method in [14] is an improved version of the scaled peak cancellation (SPC). The SRPC increases convergence rate to the desire threshold level by using several stages of the SPC technique, which is helpful to limit the peak re-growth caused by the filter. Let the u in (3,23) be the input signal of next stage, the i^{th} stage SRPC can be described as follows.

$$\begin{aligned}
\mathbf{u}_1 &= \mathbf{u} - \alpha_1 \cdot \mathbf{p}_{f1} \\
\mathbf{u}_2 &= \mathbf{u}_1 - \alpha_2 \cdot \mathbf{p}_{f2} \\
&\dots \\
\mathbf{u}_i &= \mathbf{u}_{i-1} - \alpha_i \cdot \mathbf{p}_{fi}
\end{aligned}
\tag{3,24}$$

The scale factors α_i is given by

$$\alpha_i = \frac{\max(|\mathbf{p}_i|)}{\max(|\mathbf{p}_{fi}|)}
\tag{3,25}$$

The SRPC technique will be used later in this thesis to reduce the PAPR of the generated OFDM signal. The clipping threshold need to be decided before applying the SRPC. By observing the distribution on the absolute value of the normalized OFDM baseband data, as shown in Fig. 3.16, most of the sample amplitude is below 0.5, so it is justified to set the clipping threshold to a little bit higher than 0.5 which is 0.6 in this work.

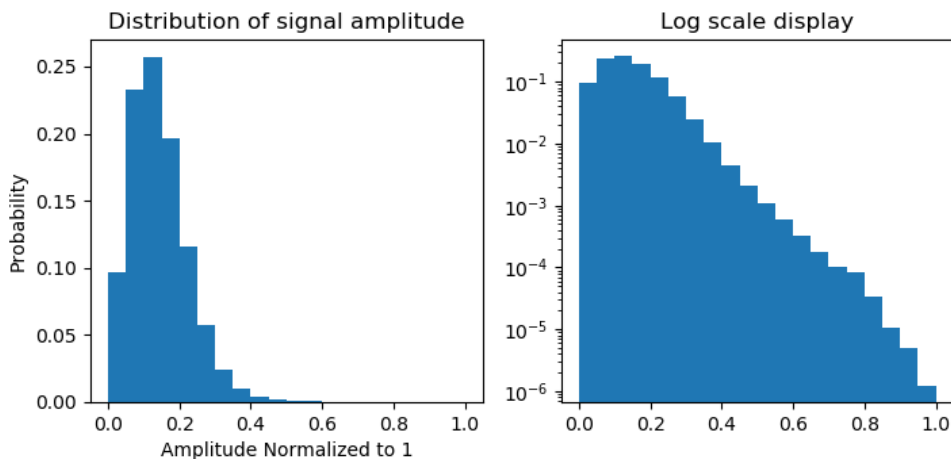


Fig. 3.16 Distribution of generated signal amplitude

Low-pass filtering is required by the SRPC, so a FIR filter was designed. In the proposed OFDM system, the lower 56 subcarriers of 64-IFFT are used, the low pass filter should pass the lower part of 87.5% of the bandwidth. Besides, three iterations ($i = 3$) are specified for the SRPC. Fig. 3.17 shows the comparison of the original signal and the signal after applying the clipping and filtering SRPC CFR method. It can be observed that the power spectrum is not significantly affected, and the amplitude after clipping and filtering is adequately controlled at the threshold of 0.6.

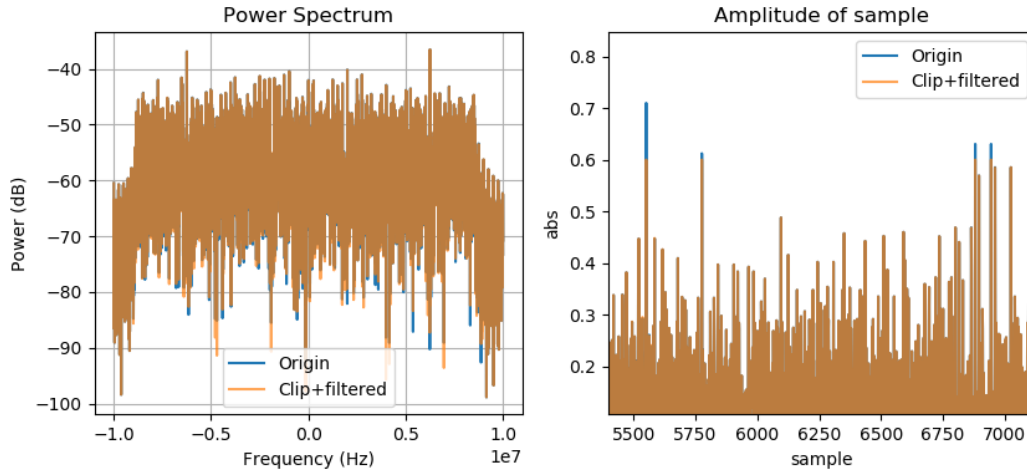


Fig. 3.17 Spectrum and Amplitude comparison on CFR signal

From the spectrum/time perspective, it is hard to tell the improvement regarding the PAPR. In order to know the improvement of the overall PAPR performance, the PAPR for every symbol sample signal is calculated individually, and the probability distribution function (PDF) and the cumulative density function (CCDF) of both the original signal and the signal after clipping and filtering were calculated. As shown in Fig. 3.18, the PAPR of the original baseband data is around 11-16 dB and approximately following the Gaussian distribution, with around 60% of the possibility to exceed 13 dB. The overall PAPR level is lower after the clipping and filtering. The mean PAPR has been moved left to about 11 dB. The peak probability is higher, and the shape is narrower, which means more likely to send the signal at the average level.

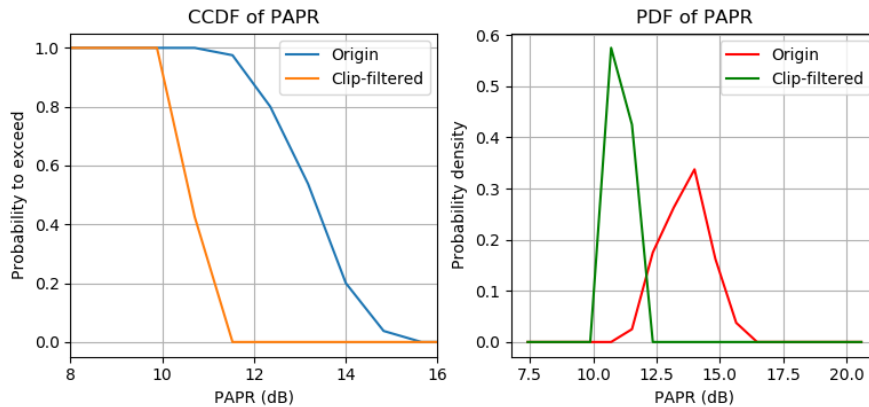


Fig. 3.18 CCDF & PDF of PAPR with or without clipping and filtering

3.2.7. Channel simulation

The UAS communication system has the same characteristic of a mobile wireless system, i.e., both operate in a time-variant communication channel. The output signal after the communications channel is described by

$$y(t) = \int_{-\infty}^{\infty} x(t - \tau)h(\tau, t)d\tau \quad (3,26)$$

where $h(t)$ is the time-variant impulse response of the channel. The large-scale attenuation model and the small-scale fading model can be used to simulate the UAS communication channels [12]. The large-scale attenuation model is suitable for UAS in the open environment where the attenuation of the signal in free space and the transmit path attenuation are considered, while multipath fading situation are not included. The impulse response $h(t)$ for the large-scale attenuation model can be finally determined by the distance between the receiver and the kinematic position such as the angle of the UAS. The observed impact when considering a small burst of signal in our simulations are a constant phase shift, a small delay and the power loss on the whole original signal. The small-scale fading model is mainly for cities or mountain environments, since it takes into account the multipath fading factors. The impact of the multipath channel is that the received signal will be the sum of several signals coming with different amplitudes and delays (resulting from the different paths), which can be simulated by performing the convolution of the input signals with the impulse response $h(t)$ of a multipath fading system. This will cause the signal to have different phase-shifts and magnitude changes on different frequencies. At the end, the channel simulation was simplified and we used a fixed impulse response with additive white Gaussian noise to meet a certain level of signal to noise ratio (SNR). In addition, a constant time shift τ was added to represent the delay.

$$y[n] = \sum_{m=-\infty}^{\infty} x[n - m - \tau]h[m] + AWGN(SNR) \quad (3,27)$$

Without any kind of compensation, the received signal will be significantly distorted and thus, the demapped QAM constellation will be blurry and will introduce a huge amount of bit errors. As shown in Fig. 3.19, the demapped constellation is tremendously distorted and the image can be hardly recognized with a byte error rate of 99.6% and a bit error rate of 49.83%.

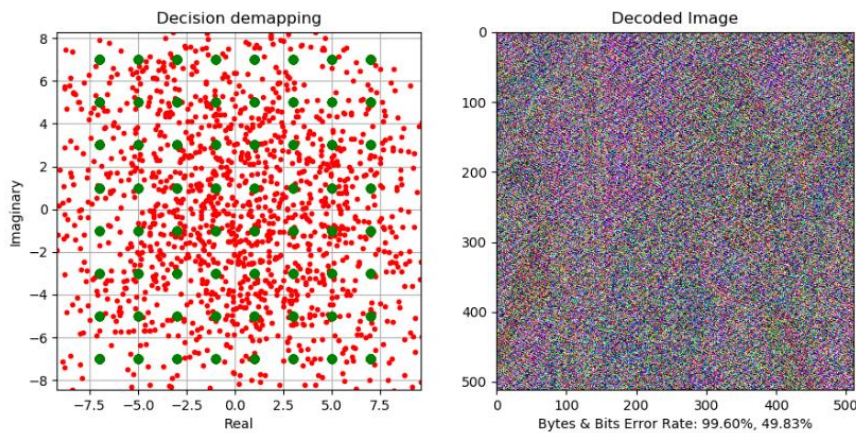


Fig. 3.19 Constellation decode without synchronization and equalization

3.2.8. Synchronization

Synchronization techniques are essential for OFDM-based communications to compensate for the symbol time offset (STO) and the carrier frequency offset (CFO) of the received signal. The STO is caused by the signal transferring time in the space and the processing time on both the transceiver and the receiver. If the FFT window begins in the cyclic prefix or the window begins after the starting point of the data part, inter-symbol interference (ISI) will be introduced and it cannot be compensated by using equalization. The CFO can be caused by the Doppler effect as the UAS is moving or by the slight frequency difference of the local oscillator (LO) between the receiver down-converter and the transmitter up-converter. Depending on the relative velocity of the UAS to the GCS, and the precision of the LO frequency, the impact of the CFO is different. The OFDM system relies on the orthogonality of subcarriers to avoid inter-carrier interference (ICI), however, the CFO misleads the receiver to calculate the FFT and generates a shifted spectrum which favours the appearance of ICI. Usually, the synchronization will first compensate the STO to find out the beginning of an OFDM symbol and then it will correct the CFO with the time-aligned signal.

The starting point of OFDM symbols can be accurately determined by estimating the STO with a synchronization technique at the receiver. The estimation of STO can be implemented in both time-domain and frequency domain. In the time-domain, STO can be estimated by using the cyclic prefix. For example, using the sliding windows method and considering two windows spaced N samples apart (where N is the length of the data symbol), where each has the same width as the CP. The sliding windows move through the received signal and calculating the similarity by summing the difference of both signals for different delays. Since the CP is a copy of a part of the data signal, the index of the minimum distance results at the starting point of an OFDM symbol. Taking into account the received signal y , the STO can be found by searching the point where the difference between two blocks of M samples within two sliding windows is minimum [10], as shown below.

$$\hat{\delta} = \underset{\delta}{\operatorname{argmin}} \left(\sum_{i=\delta}^{M-1+\delta} |y[n+i] - y[n+N+i]| \right) \quad (3,28)$$

By applying this CP-based sliding windows method to the simulation signal distorted by the multipath channel model, the STO compensation result is shown in Fig. 3.20. The starting point can be well estimated, located in the valley of the “STO-estimation” axis.

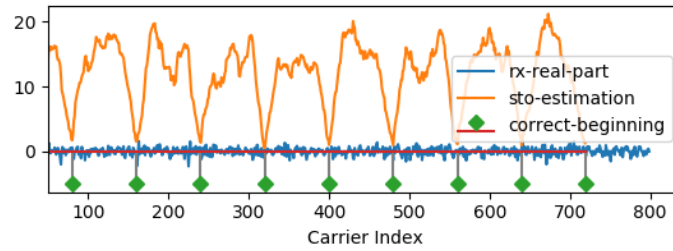


Fig. 3.20 STO Estimation using CP minimum difference method

The other time-domain STO estimation method is achieved by adding a training symbol before a set of symbols. The training symbols, also known as the preamble [15], are special symbols that are known by the receiver. The beginning of the symbol can be estimated by a single sliding window that calculates the correlation. It differs from the former method because involves preambles, which will downgrade the maximum data throughput. However, it does not suffer from the effect of the multi-path channel and it only requires one sliding window. Apart from the time-domain methods, frequency-domain based STO estimation methods [16] are also available since the phase rotation is proportional to subcarrier frequency.

For CFO estimation, like the STO, methods can also be based on the time-domain or frequency-domain and make use of the CP or the preamble. For example, the time-domain method that uses the CP, assuming a good enough symbol synchronization, and the location of CP in the data symbol is already known, the CFO can be found from the phase angle of the product of CP and the corresponding part of the data symbol. Denoting N_{cp} as the length of CP, N the OFDM IFFT length and y^* is the complex conjugate of the received signal y , the CFO can be estimated as below.

$$\hat{\varepsilon} = \frac{1}{2\pi} \arg \left(\sum_{n=1}^{N_{cp}} y^*[n]y[n+N] \right) \quad (3,29)$$

The argument operation results to the angle from $[-\pi, \pi)$, hence the estimated $|\hat{\varepsilon}| < 0.5$. It means that a big CFO ($|\hat{\varepsilon}| \geq 0.5$) cannot be estimated by this CP-based technique. Other CFO estimation techniques based on the training symbol can be used to increase the CFO estimation range. For example, the frequency-domain approach by Moose in [17], first creates a training symbol and repeats it K times and then inserts it into the data symbols. This technique can increase the CFO estimation range by K times, but at the price of sacrificing the MSE. We tested the CP-based CFO estimation with the generated simulation data, as shown in Fig. 3.21, but since no deliberate frequency offset was added, the values were all close to 0.

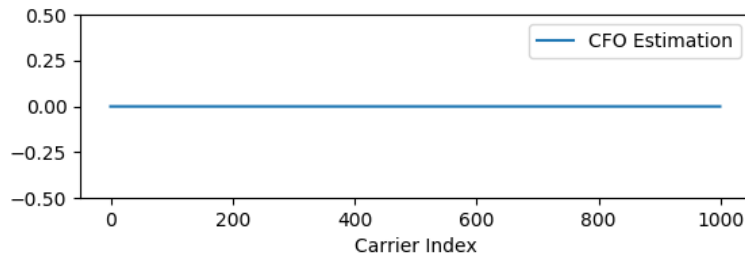


Fig. 3.21 CP-based CFO estimation with simulation data

3.2.9. Channel estimation

With the help of pilot subcarriers, the receiver can estimate the channel by comparing the received signal with the known pilot value. Comb-type pilots were used in the signal generation in order to be able to estimate the fast fading channel. As shown in Fig. 3.22, before the channel estimation, the receiver should have done synchronization, find out the start of the symbol and have removed the CP. After applying FFT to every OFDM symbol batch, according to the Comb-type arrangement, the pilots can be identified.

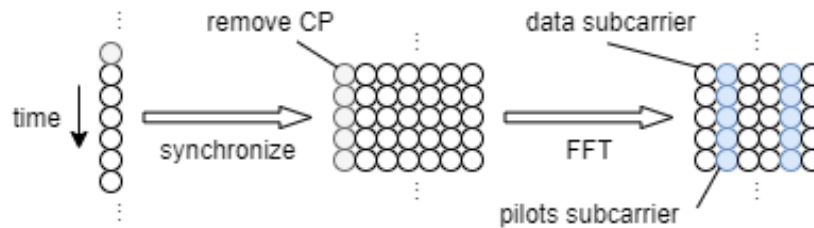


Fig. 3.22 Procedure for pilots extraction

The effect of the channel can be estimated for each frequency component by performing interpolation to the pilot subcarriers extracted after the FFT operation at the receiver. Popular interpolation methods include linear interpolation, second-order polynomial interpolation, and cubic spline interpolation. High order interpolation introduces unnecessary complexity to the estimation, while linear interpolation is enough and thus more suitable since is less computational complex for this simulated scenario. In Fig. 3.23, the two plots on the top show the interpolated result of the first OFDM symbol using different interpolation orders; the two plots on the bottom are the magnitudes and phases of the first 100 symbols at the frequency of the four pilot carriers. The multipath channel simulation fairly changed the phases and amplitudes with the same scale, the pilot value should not change over time. However, the FIR filtering after the peak cancellation made the equalization scales different between samples.

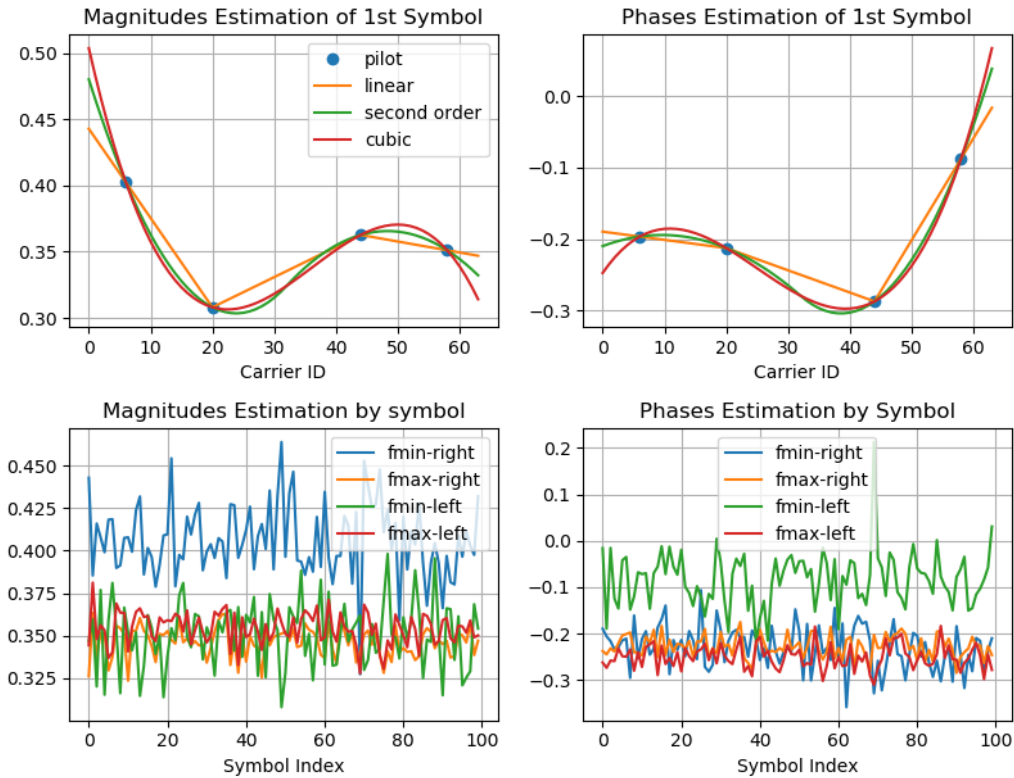


Fig. 3.23 Channel estimation results

Let's denote the channel estimation made with an OFDM symbol \hat{H} , while R_t is the received OFDM symbol after FFT, then, the equalized signal \hat{Y}_t is obtained after a simple division [10]. In the real-time implementation, the OFDM symbol at moment (t) is calculated by the estimation at moment ($t - 1$) to reduce the lagging.

$$\hat{Y}_t[k] = \frac{R_t[k]}{\hat{H}_{t-1}[k]} = \frac{X_t[k]H_t[k]}{\hat{H}_{t-1}[k]} \approx X_t[k] \quad (3,30)$$

After the channel equalization, most constellation points were equalized and reallocated to the right position in the constellation. The recovered image in Fig. 3.24 is pretty accurate with only 3.08% of bytes error. By comparing it with Fig. 3.19 that showed 99.6% of bytes error, we can conclude that the equalization made a significant improvement. Inserting more pilot subcarriers or considering higher-order interpolation methods could improve the channel estimation and thus the equalization.

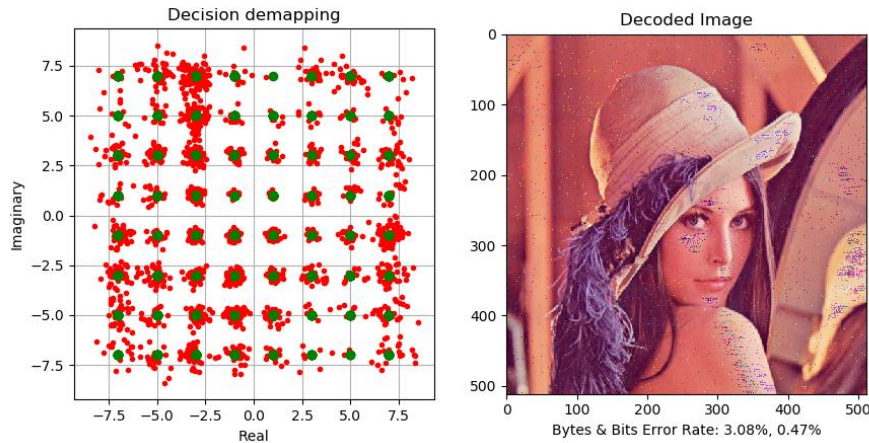


Fig. 3.24 Constellation and decoded image with pilot equalization

3.3. Techniques to improve power efficiency

As discussed in subchapter 3.1.3, the proposed communication system automatically changes the TX power according to the distance and the received signal strength. To adjust the output power, there are some options in the RF power amplifier circuit that are available.

Before looking at the options, it is necessary to introduce and discuss the topic of RF power amplifiers to know the basic principles of amplification and some interesting properties. Fig. 3.25 shows a simplified block diagram for an amplification system for communications: from the analog-to-digital converter to the PA, passing through the upconversion to RF and pre-amplification. Thus, it requires at least four stages to generate the RF signals and send through the air with a certain output power. Since the PA cannot directly amplify the small signal to high power, one or more pre-amplifiers are required to increase the power step by step. Sometimes, the variable gain amplifier (VGA) might exist in the pre-amplify stage in order to give the flexibility to set the power. In this general scheme, it is assumed that both the pre-amplifier and the PA are ideally linear and thus the gain value can be added without considering any gain compression. For example, the DAC outputs a signal with 0 dBm mean power, the VGA together with the Pre-amplifier can add a gain between 10 to 20 dB, and the PA can add 13 dB of gain. The system can finally present a signal with 23 to 33 dBm of mean output power.

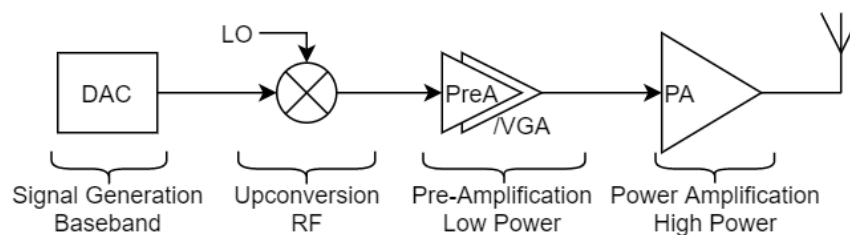


Fig. 3.25 General power amplifier scheme

Apart from using the VGA in the pre-amplification stage to control the output power, every stage can have its way to affect the output power. The DAC can control the power by giving small digital numbers, which is easy to handle and low-cost, however in a communication system, this will decrease the DAC dynamic range and thus degrade the quality of the generated signal due to the quantization noise. Another possibility is to tune the output power by changing the PA supply, by varying the drain voltage or the gate bias, which affects the Q-point of the transistor. By setting the position of the transistor Q-point, the amplifier will operate in different classes that have differences in efficiency and linearity.

If the PA is not operating in the linear zone, the output power cannot be easily calculated by adding a gain, the PA gain starts compressing and unwanted nonlinear terms appear at the PA output. On the other hand, the power efficiency is higher when the PA is operated close to saturation, i.e., in the nonlinear zone. If the PA is not operated in hard saturation, the nonlinear distortion can be compensated by using linearization techniques such as the digital predistortion (DPD). Furthermore, the required linearity depends on the type of modulation used in the communication system. The nonlinear distortion introduced by the PA affects the quality of the received signal, degrading the SNR and causing bit errors in the receiver. Frequency-based and phase-based modulation methods such as BFSK, FSK and QPSK, require lower SNR than those modulation methods based on the amplitude modulation such as AM and QAM, either using single carrier or multicarrier (e.g., OFDM) approaches. Fig. 3.26 was presented in [18] and shows the bit error rate (BER) performance analysis of image transmission using various modulation schemes with and without OFDM. As shown in Fig. 3.26 and Table 3.5, with OFDM the required SNR is slightly less stringent than with non-OFDM. Therefore, for example, with a 64-QAM OFDM signal, at least 22.8 dB of SNR (considering only a noisy channel) is necessary ensure 0.1% of BER for the image transmission system.

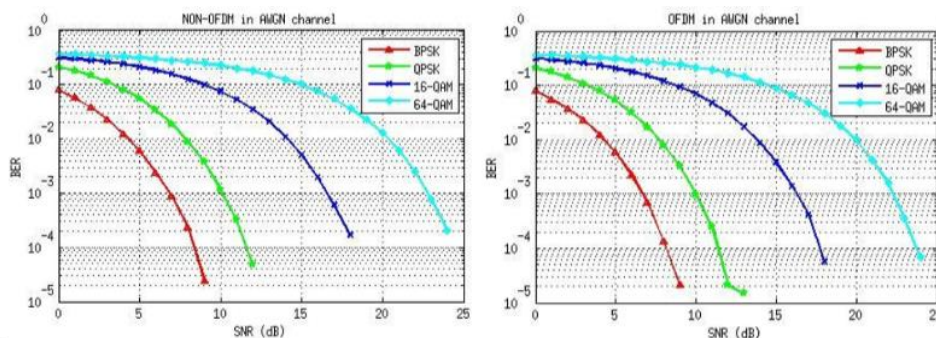


Fig. 3.26 SNR vs BER in different modulation schemes [18]

Table 3.5. SNR for BER at 0.1% in AWGN channel [18]

Modulation Method	Modulation Scheme			
	BPSK	QPSK	16QAM	64QAM
SNR for Non-OFDM	7 dB	10.5 dB	16.8 dB	23 dB
SNR for OFDM	6.8 dB	10 dB	16.3 dB	22.8 dB

The solution proposed in this thesis to improve the power efficiency consists by finding a suitable polarization for the PA to trade-off linearity and efficiency and then operating the PA as close to saturation as possible. DPD linearization will be included to guarantee the required linearity levels, evaluated in terms of adjacent channel power ratio (ACPR) and normalized mean square error (NMSE). The latter figure of merit (FoM) has a direct relation to the FoMs at the receiver, i.e., the SNR and BER.

3.3.1. PA bias point selection and operating classes

Amplifiers can be classified into different classes according to the bias point configuration at which the PA operates. Therefore, the power transistor class of operation is defined by the fraction of the RF cycle over which the power transistor conducts. For example, a class-A amplifier that operates the Q-point in the linear portion of its characteristic curves, shows the best linearity, since the output signal conducts a phase shift of 2π (100% conduction angle), but presents the lowest efficiency (ideally around 30%). Table 3.6 shows the existing trade-off between linearity and efficiency according to the power transistors operation class [19].

Table 3.6. Amplifier classes and properties

Class	Description	Conduction Angle	Ideal Efficiency	Linearity
Class A	Using the mid-point of the linear range as the Q-point, providing RF drive signal never exceeds the boundary values.	2π	30%	Best
Class B	Zero quiescent current, and half of the waveform for each half of the bipolar.	π	50%	Crossover distortion
Class AB	The Q-point has a little crossover the class B.	$(\pi, 2\pi)$	50-60%	Similar to Class B
Class C	Conduct for less than half cycle of the input signal to achieve higher efficiency.	$(0, \pi)$	Around 80%	Heavy distortion
Class D~F	Operating at switch mode, with the highest efficiency but also highest non-linearity.	0	100%	The worst distortion

Although the power efficiency figures may change when considering modulated signals with high PAPR, the aforementioned trade-off that arises when considering different polarization configurations are essential references for

engineers to select or design a proper PA that fits with their specific application. In this thesis, to increase the power efficiency but not cause heavy distortion, we will use a class-E PA, that can operate in switched mode or linear mode (class B, AB) depending on the polarization used and the input power of the signal.

3.3.2. Overview of commercial power amplifiers

In commercial amplifiers, unlike ideal amplifiers, the behavior and power efficiency may depend on several features, such as the material or process used, the design of the impedance matching circuit for the input and output port, the bias circuit, etc. Designing an RF power amplifier is not the objective of this thesis, but the considerations regarding the PA selection are worth mentioning. This work supposes that the final product is a payload which can be equipped into a UAS, thus a PA system suitable for the UAS will be recommended.

In recent decades, with the development of semiconductor and material, transistors have gone through several generations. LDMOS transistors were first introduced for high power amplifiers that offered higher gain and better back-off linearity compared to the bipolar transistors. Later, Gallium Arsenide (GaAs) devices appeared and have long been reported as the replacement to the LDMOS, since they can be used at higher frequencies. Nowadays, Silicon Carbide (SiC) and Gallium Nitride (GaN) transistors are thought to be the chosen ones for high-power amplifier design. GaN devices have properties such as high electron mobility, high current gain and cutoff frequency (f_T), large power handling capability and allow high-temperature operation, which make them more suitable to operate in higher power and frequency than the GaAs devices. In the early years, for narrowband and low-frequency cases, LDMOS high power transistors were the technology of choice for wireless communications because GaAs and GaN required more advance linearization techniques than LDMOS [20]. However, since the linearization techniques such as DPD nowadays have improved a lot, GaN devices have gained momentum and are present in many communications devices, from base-stations to handsets. Fig. 3.27, taken from [21], shows the available selection of transistors' technology for different power and frequency.

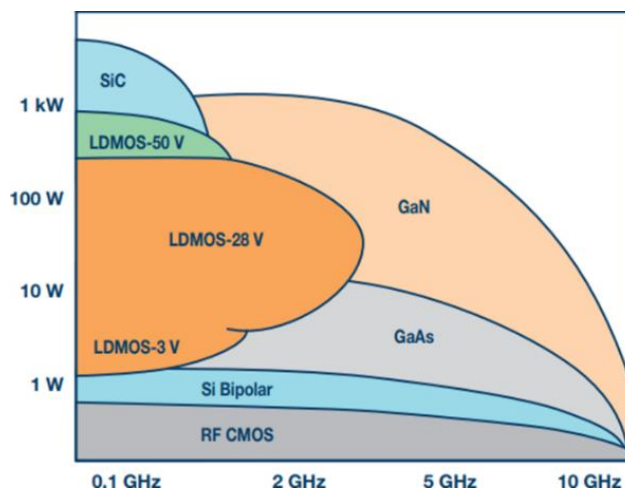


Fig. 3.27 Power vs frequency of different transistor technologies [21]

When standing in the view of making a product for the UAS payload for wireless communications, apart from the signal quality, the most critical specification are the size and the weight. The available space in the UAS is limited, and weight affects the flying performance. There is no doubt to use PA which is small and less weight. Fortunately, the latest commercial GaN PAs have smaller size compare with LDMOS, also the property of allowing high-temperature operation ease the design of heat dissipation and hence decrease the weight. The only concern now for using the GaN devices is to guarantee linearity levels by using linearization techniques, which is the topic that will be addressed in the following subchapters.

3.3.3. Digital predistortion techniques

The digital predistortion (DPD) linearization technique has become one of the most widely used solutions for coping with the trade-off between linearity and efficiency in PAs. DPD algorithms are based on the behavioral modelling of the power amplifier. Behavioral or black-box models are not physical or circuitual models that take into account the constitutive relations of voltages and currents along the PA components. Instead, they are parametric mathematical models describing the nonlinear behaviour and memory effects of the PA extracted from simple input-output data observations. The performance of the model will degrade if the working environment changes a lot. For example, in this thesis, the TX gain is adjusted automatically according to the specific distance between the UAS and the GCS, and consequently the voltage supply of the PA may also change. Therefore, linearity performance of a trained DPD will decrease if the behavioural model is not adapted to the new environment.

Many commonly used polynomial-based behavioral models can be seen as simplified approximations of the general Volterra series. The discrete-time low-pass equivalent Volterra series formulation considering complex signals is described in the following.

$$\hat{y}[n] = \sum_{p=1}^P \sum_{q_1=0}^{Q-1} \sum_{q_2=q_1}^{Q-1} \dots \sum_{q_p=q_{p-1}}^{Q-1} \dots \sum_{q_{2p-1}=q_{p-1}}^{Q-1} h_{2p-1}(q_1, q_1, \dots, q_{2p-1}) \prod_{i=1}^p x[n-q_i] \prod_{j=p+1}^{2p-1} x^*[n-q_j] \quad (3,31)$$

The series is composed by $2P-1$ kernels of increasing dimensional order. The main drawback of using the full Volterra series is that the number of parameters grows exponentially when considering higher order kernels, which requires a powerful computer to handle and it is not friendly for low-cost implementations.

Popular simplified versions of the Volterra series are memory polynomial (MP) [22] and generalized memory polynomial (GMP) [23] behavioural models presented in the last decades. MP is the simplest model that only takes into

account the memory effects and its powers, but without considering cross-memory terms.

$$\text{MP:} \quad \hat{y}_{MP}(n) = \sum_{p=0}^{P-1} \sum_{m=0}^{M-1} a_{p,m} x(n-m) |x(n-m)|^p \quad (3,32)$$

The GMP instead, includes cross-memory products (lagging and leading terms) which allows capturing more nonlinear memory modes of the PA behavior. When comparing both models with the same numbers of coefficients, the GMP usually outperforms the MP (for example in [24]), at the price of introducing more computational complexity when targeting hardware implementation.

$$\begin{aligned} \text{GMP:} \quad \hat{y}_{GMP}(n) = & \sum_{k=0}^{K_a-1} \sum_{l=0}^{L_a-1} a_{k,l} x(n-l) |x(n-l)|^k \\ & + \sum_{k=1}^{K_b} \sum_{l=0}^{L_b-1} \sum_{m=1}^{M_b} b_{k,l,m} x(n-l) |x(n-l-m)|^k \\ & + \sum_{k=1}^{K_c} \sum_{l=0}^{L_c-1} \sum_{m=1}^{M_c} c_{k,l,m} x(n-l) |x(n-l+m)|^k \end{aligned} \quad (3,33)$$

There are plenty of other behavioral models in literature, some of them implemented as the combination of piecewise functions instead of polynomials, such as the decomposed vector rotation (DVR) model in [25]. The DVR is a model derived from a modified form of canonical piecewise-linear (CPWL) function, and as shown in [26] presents similar linearization performance as GMP, but with better numerical properties.

The estimated output vector of a general behavioural model $\hat{\mathbf{y}}$ can be described as the product between a data matrix \mathbf{X} that contains the basis functions or components, and a vector $\boldsymbol{\omega}$ that represents the parameters of the model.

$$\hat{\mathbf{y}} = \mathbf{X}\boldsymbol{\omega} \quad (3,34)$$

For example, in the MP model, the \mathbf{X} contains all the memory and power terms of the input signal, and the $\boldsymbol{\omega}$ is a vector containing all the $a_{p,m}$ parameters.

The estimation error \mathbf{e} is defined as the difference between the measured PA output \mathbf{y} and the modelled PA output $\hat{\mathbf{y}}$.

$$\mathbf{e} = \mathbf{y} - \hat{\mathbf{y}} = \mathbf{y} - \mathbf{X}\boldsymbol{\omega} \quad (3,35)$$

The objective is to find the values of the $\boldsymbol{\omega}$ parameter that can minimize the error. Despite being a nonlinear model, it is linear in parameters, so linear regression analysis can be used. Therefore, taking the l_2 -norm squared of the identification error, the LS problem is defined as follows.

$$\min_{\omega} \|e\|_2^2 = \min_{\omega} \|y - X\omega\|_2^2 \quad (3,36)$$

Taking the derivative of the cost function $J(\omega) = \|e\|_2^2$ and setting it to zero, the solution to the LS problem is shown as below, where the H denotes the conjugate transpose.

$$\omega = (X^H X)^{-1} X^H y \quad (3,37)$$

Now, let us focus how to extract the DPD parameters instead of the parameters for the PA behavioural model. The block diagram of a closed-loop DPD following a direct learning approach is shown in Fig. 3.28. The coefficients are directly updated according to the measured outputs from the PA device and then applied to the DPD. To ensure that the feedback signals have the same nature as the input signals, time alignment and rotation compensation are applied to the feedback data before pushing into the adaptation of the coefficients.

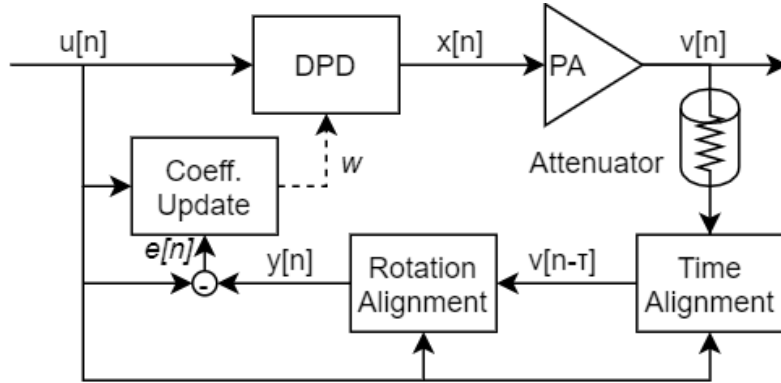


Fig. 3.28 Direct learning approach DPD diagram

The DPD block in the diagram is the behavioral model that takes $u[n]$ and calculates the predistorted signal $x[n]$ using the DPD coefficients w . The model can be MP or GMP, or other effective behavioral models. The data sequence after DPD are digital-to-analog converted and upconverted to the RF frequency of operation and then sent to the PA. The PA output signal goes through a linear attenuator to make sure not to exceed the maximum amplitude at the ADC. The transfer time on the circuit and delays on the feedback path introduce an unknown latency, which needs to be compensated by the time alignment block. Since the input data is known by the system and the distorted data remains similar to the original data, the time shift can be estimated by the max index of their correlations. To get rid of the phase and the gain differences, normalization and absolute operations were performed before calculating the time shift. Also, to increase the precision, up-sampling is used to allow fractional delay. The mathematic expression to find the time shift by correlation is shown as follows.

$$\hat{\tau} = \underset{\tau}{\operatorname{argmax}} \left(\sum_{n=0}^N |u[n]| |v[n + \tau]| \right) \quad (3,38)$$

The phase response of the PA also needs to be cancelled out. With the time-aligned feedback signal, the fine phase shift can be estimated by rudely applying all the possible phase shifts and then choosing the best match. The expression to estimate phase shift φ is shown as below, where \hat{t} is the estimated time shift.

$$\hat{\varphi} = \min_{\varphi} \left(\|u[n] - v[n + \hat{t}]e^{i\varphi}\|_2^2 \right), \quad \varphi = 0, \dots, 2\pi \quad (3,39)$$

In the end, the distorted signal $y[n]$ can be obtained by applying the estimated time shift \hat{t} and the phase shift $\hat{\varphi}$ as below. If the $v[n]$ has been up-sampled during the estimation, it should be down-sampled after the time alignment and rotation compensation.

$$y[n] = v[n + \hat{t}]e^{i\hat{\varphi}} \quad (3,40)$$

For testing purposes, when these compensations have been applied to the received signal, the synchronization and equalization of the OFDM signal are no longer necessary. The closed-loop error is defined as follows (note that the y and u have been normalized to the same level).

$$e[n] = y[n] - u[n] \quad (3,41)$$

The PA behavioral models or basis functions can also do well in modelling the PA inverse response. Let's denote \mathbf{U} as the data matrix containing the basis functions of a specific behavioural model and \mathbf{w} as the vector of parameters. The data matrix \mathbf{U} has dimensions $L \times M$, where L is the number of samples ($n = 0, 1, \dots, L - 1$) and M is the number of basis functions. The complex vector \mathbf{w} has the length of M , and the output vector \mathbf{x} has the length of L . The output of the DPD \mathbf{x} can be described as follows.

$$\mathbf{x} = \mathbf{u} - \mathbf{U}\mathbf{w} \quad (3,42)$$

The coefficients can be extracted iteratively following a gradient descent approach as shown below,

$$\mathbf{w}_{j+1} = \mathbf{w}_j + \mu(j)\Delta\mathbf{w} \quad (3,43)$$

Where j denotes the epoch number and μ is the learning rate function that can be a constant or change according to the epoch number. The coefficients increment $\Delta\mathbf{w}$ is obtained following the LS solution as

$$\Delta\mathbf{w} = (\mathbf{U}^H\mathbf{U})^{-1}\mathbf{U}^H\mathbf{e} \quad (3,44)$$

After a few epochs the value of the coefficients in \mathbf{w} should converge to a steady state value.

3.3.4. The power amplification testbed

A PA testbed was built for conducting transparent, replicable, and remotely operable testing. It comprises a Pluto-SDR, a Pre-Amplifier, a PA, a power supply, and three necessary attenuators.

The Pluto-SDR [27] is a software-defined radio platform, capable of sending/receiving RF analog signals from 325 to 3800 MHz, at up to 61.44 Mega samples per second (MSa/s) with channel bandwidth of 20 MHz. The Pre-Amplifier is a high linearity class-A amplifier capable of increasing the signal power without losing signal quality. The PA which was designed in [28] is a class-E power amplifier capable of delivering high-efficiency amplification at 2.4 GHz. The attenuators were inserted at the TX and RX port of the Pluto-SDR to guarantee that the TX/RX signal power will not exceed the maximum thresholds and protect the devices. The power supply and the Pluto-SDR are connected to a host PC, which runs Matlab scripts to provide a remotely operable bridge to Python. Fig. 3.29 shows the hardware of the testbed and its block diagram.

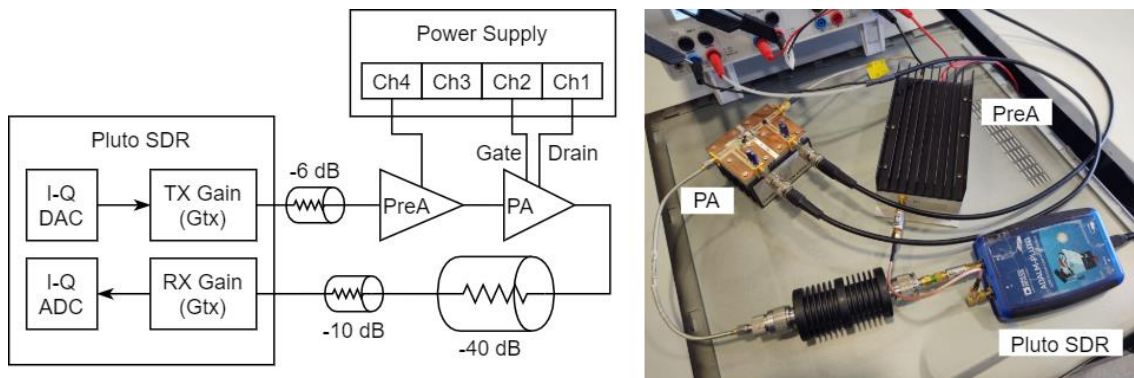


Fig. 3.29 Block diagram and photograph of the testbed

With this testbed, it is possible to control variables of TX gain, RX gain, drain voltage, and gate voltage. The following subchapter will characterize the PA performance by applying different values to these variables and testing with the filtered OFDM signals which have been generated in subchapter 3.2.6. Before doing the tests, the received signal power of Pluto-SDR was calibrated by finding out the difference between the calculated root mean square (RMS) power and the standard RMS power with an RF power meter.

The input power sweep simulation of the PA was given in [28]. With the gate voltage of -3.5 V and drain voltage of 28 V, the corresponded efficiency, gain, and output power are shown in Fig. 3.30. The PA has the potential to deliver up to 83.4% of efficiency when feeding the PA with a 30 dBm input signal at 2.4 GHz. However, the efficiency degrades rapidly when the input power is decreased. The gain also changes with the input power, so the output power is not proportional to the input power, which implies that the PA has to be characterized to build the correct table of specific output power levels. With the testbed described in Fig. 3.29, the maximum input power for the PA is limited to 19 dBm, so the maximum

efficiency it will be able to reach is around 50% and the maximum output power is lower than 40 dBm.

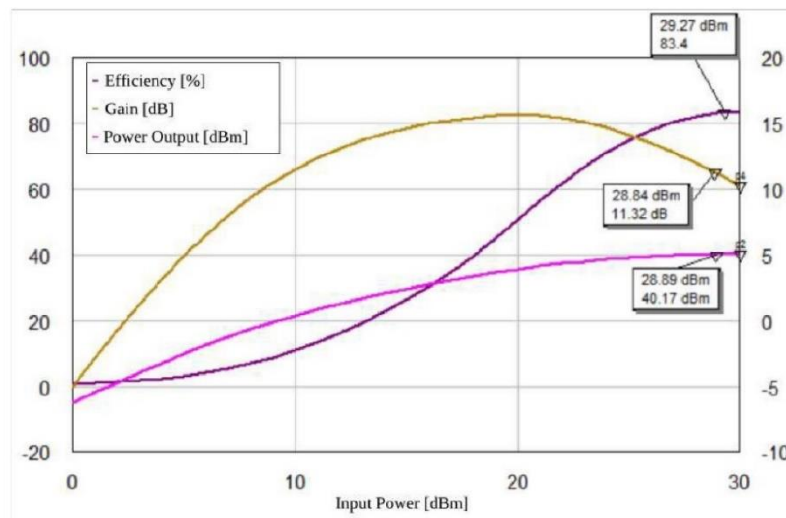


Fig. 3.30 Input power sweep simulation of the PA [28]

3.3.5. Searching for the optimal PA configuration

To save the dynamic range of the DAC/ADC, it is better to change the TX/RX gain inside the Pluto-SDR instead of sending small digital numbers. Assuming the linear case, when the TX gain increase 1 dB, the RX gain should decrease 1 dB and vice versa. Once the power of input signal changes, the power efficiency and linearity performance are different.

To maintain the power efficiency, the gate and the drain supply voltage of the PA have to be adjusted properly, consequently the optimal voltage configurations have to be tested out. Besides, the DPD coefficients are adapted for a PA assuming the static environment, which means that the change of environment (gate and/or drain voltage) will degrade the DPD linearization performance and might even make it worse than without the DPD. There are some works that increase the PA power efficiency by including dynamic supply modulation techniques. For example, the works on envelop tracking and dynamic biasing of the PA in [29], show that for transmitting different mean output power values, it exists an optimal voltage configuration that maximizes power efficiency.

The objective is then to find the optimal configuration of TX gain (G_{tx}), RX gain (G_{rx}), gate voltage (V_{gs}), and drain voltage (V_{dd}) for the PA to be able to transmit different levels of output power (P_{out}) with maximum power efficiency, which is the main purpose of the proposed automatic TX power controller. Two searching strategies can be considered. The first strategy is to send the signal with different supply voltages, TX gain settings and then perform DPD to get the table of performances, and select the optimal configuration. However, the three degrees of freedom (G_{tx} , V_{dd} , V_{gs}) and the number of iterations of the DPD make the testing time grow tremendously.

The second approach is to try different settings without DPD and take the best trade-off between power efficiency and linearity for every desired TX power level. The DPD is then applied to the optimal configuration, expecting that it can meet the linearity specifications. For example, knowing that the DPD can conventionally improve the adjacent channel leakage ratio (ACLR) by about 6 dB, and the desired ACLR threshold is -45 dB. The trade-off can be found by selecting the best power efficiency setting where the ACLR without DPD is better (i.e., lower) than -39 dB.

NMSE and ACLR are the main metrics of linearity. The efficiency is calculated by $P_{out}/P_{dc} \times 100\%$. By fixing the G_{tx} to -11 dB, Fig. 3.31 shows the PA performance with $V_{gs} \in [-3.6, -2.9]V$, $\Delta V = 50$ mV and $V_{dd} \in [10, 28]V$, $\Delta V = 2V$. As it can be observed, the NMSE and ACLR have the same trend, that is, when the PA has a good NMSE performance, it also has a good value of ACLR. But the cases with the best performance of NMSE and ACLR derive in low power efficiency figures. It also shows that with $G_{tx} = -11$ dB (inferred input power of the PA is around 13 dBm), the PA efficiency is in the range of 15% to 32%. The measured results are close to the simulation results shown in Fig. 3.30.

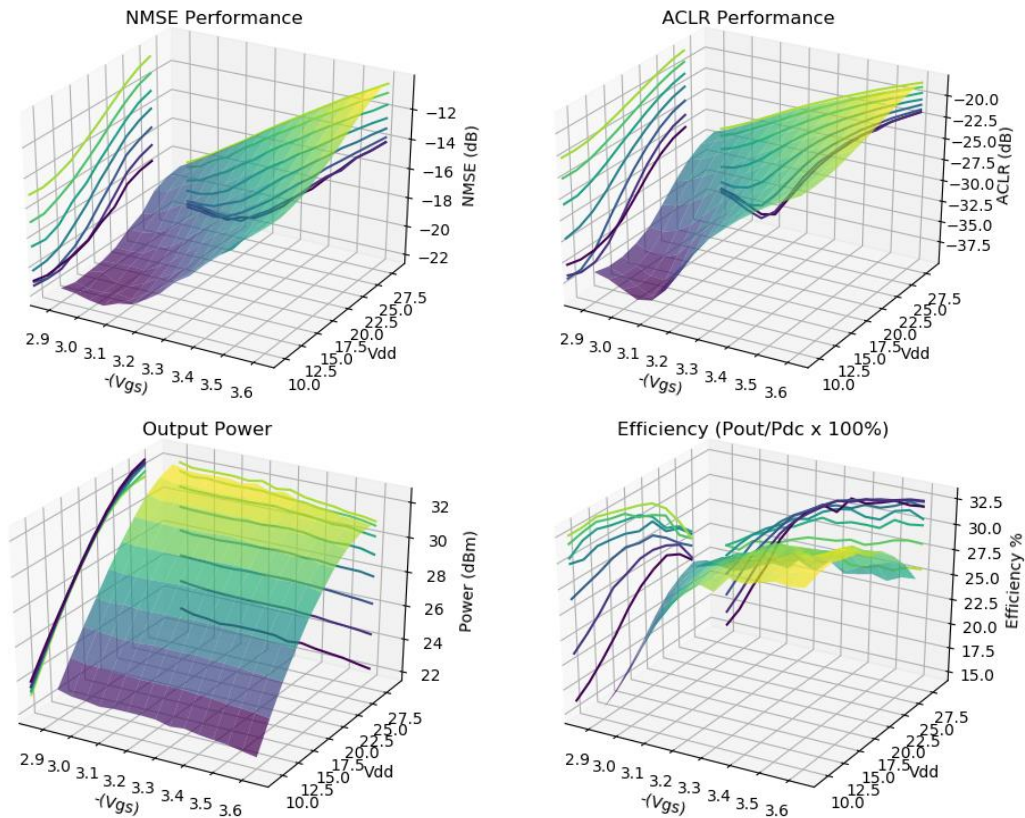


Fig. 3.31 PA performance with different supply settings without DPD

When testing with a fixed G_{tx} the optimal configuration for sending different levels of output power cannot be found. Therefore, it is necessary to scan the PA performance with different G_{tx} values. Keeping the searching range of V_{dd} , V_{gs} as

defined before, and specifying $G_{tx} \in [-5, -23] \text{ dB}$, $\Delta G = -3 \text{ dB}$, the tests resulted in a data table with 1050 rows. Apart from the three control variables, the data table has four columns of results for $NMSE$, $Efficiency$, $ACLR$, and P_{out} .

To build an intuitive representation on this seven-dimension large data set, Fig. 3.32 shows the parallel coordinates chart. The highlighted selections are results with $ACLR$ range below -33 dB and efficiency range below 15%. It shows that all the possible output powers are in range $[10, 25] \text{ dBm}$. The chart helps understand some basic property of the results, for example, it is easy to learn that the efficiency within the testing range has the maximum of up to 50.4% and the minimum of only 0.5%; the worst $ACLR$ is -9.9 dB while the best is -51.1 dB . Moreover, the maximum efficiency and output power are close to the sweep simulation results in Fig. 3.30.

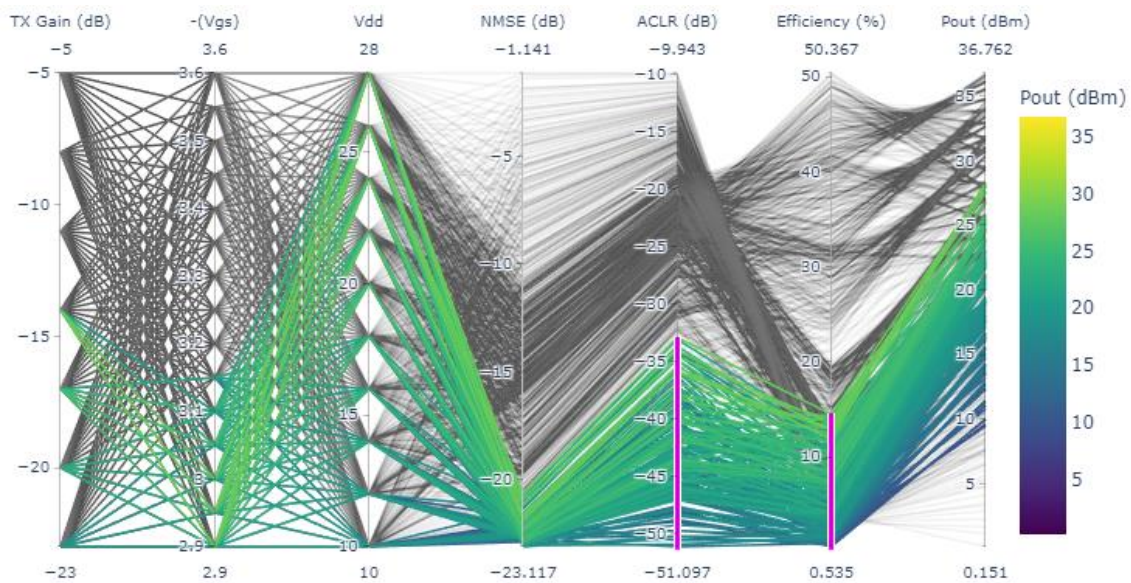


Fig. 3.32 Parallel coordinates plot of the PA configuration search results

Within the highlighted range, we defined five output power levels with gaps of roughly 3 dB. Choosing the best power efficiency for each level results in Table 3.7. The last two rows have a similar starting point of $NMSE$, $ACLR$ and output power, but the latter efficiency is higher. These configurations will be used to test with DPD in the following subchapter.

Table 3.7. The optimal configuration without DPD

TX Gain (dB)	Gate (V)	Drain (V)	NMSE (dB)	ACLR (dB)	Efficiency (%)	Output Power (dBm)
-20	-3.05	12	-22.7	-43.1	3.6	14.8
-17	-3.05	12	-22.2	-43.7	7.0	17.7
-20	-3.10	22	-22.1	-37.5	7.6	21.0
-17	-3.10	22	-21.5	-35.4	11.3	23.8
-14	-3.00	22	-21.9	-36.5	14.9	26.8
-11	-3.05	16	-21.7	-36.5	25.6	27.0

3.3.6. Perform DPD with the optimal settings

With the configurations from Table 3.7, in this subchapter we applied DPD using the same testbed to improve the linearity of the PA output signal. The GMP model described in equation (3,33) was used, with the DPD parameters of $L_a, K_a, M_b, L_b, K_b, M_c, L_c, K_c = 10, 3, 3, 2, 5, 3, 2, 5$. For each configuration, eight iterations of the LS algorithm were used to adapt the final DPD coefficients, and the results were measured three times then averaged to decrease the measurement noise. Due to the limitations in the signal bandwidth imposed by the Pluto-SDR platform, the test signal was limited to a bandwidth of 10 MHz and thus have enough margin to allocate the bandwidth expansion that appears after DPD linearization. Fig. 3.33 shows the power spectra and the AM-AM and AM-PM characteristics before and after DPD linearization, taking into account the configuration of the fifth row in Table 3.7. As expected, the DPD is able to correct the phases error and mitigate the amplitude compression, successfully reducing the ACLR up to -48.5 dB and thus meeting the threshold of -45 dB.

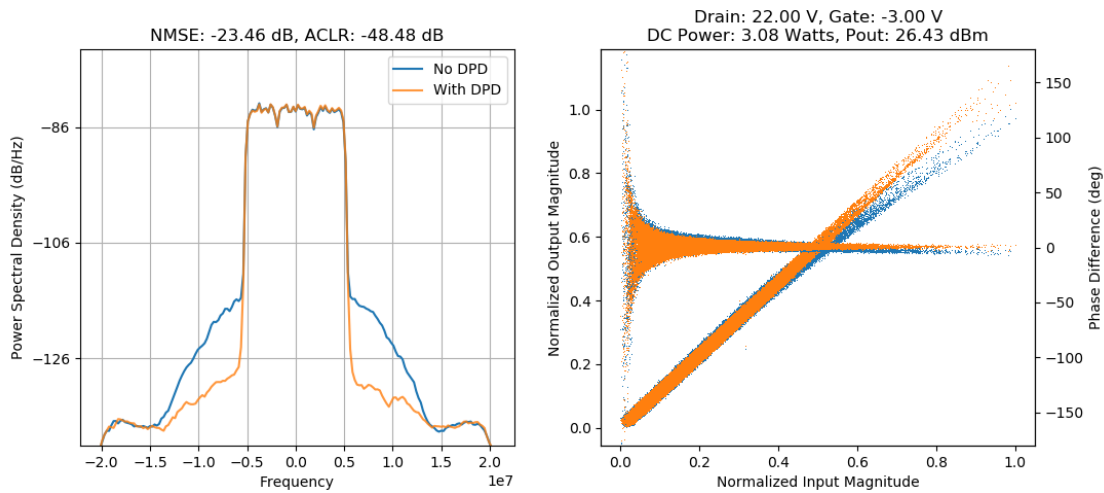


Fig. 3.33 DPD Result of the fifth optimal configuration

Table 3.8 shows the detail of the DPD results with the configurations in Table 3.7 and the fixed gate and drain configurations for comparison. The table provides a improve/degrade (\uparrow/\downarrow) impact value within parentheses. “Improve” of NMSE and ACLR means the numerical value goes down, and the “Improve” of efficiency and output power means the numerical value goes up.

As reported in Table 3.8, the DPD can improve the NMSE and ACLR performances, and does not significantly affect the efficiency and the output power, that is, the DPD can improve the linearity and maintain the efficiency performance in the optimal configurations. However, although the DPD can improve the linearity, the trade-off between efficiency and linearity still exists. So, for example, the DPD cannot meet the -45 dB of ACLR when considering the configuration in the sixth row in Table 3.8 with an efficiency of 27.3%. Instead, with DPD we can meet the configuration on the fifth row with an efficiency of 15.5%.

Table 3.8. DPD results with optimal and fixed configurations

	G_{tx} (dB)	Gate (V)	Drain (V)	NMSE (dB)	ACLR (dB)	Efficiency (%)	Output Power (dBm)
Optimal config	-20	-3.05	12	-23.1(↑0.4)	-46.3(↑3.2)	3.8(↑0.2)	15.0(↑0.2)
	-17	-3.05	12	-22.8(↑0.6)	-48.3(↑4.6)	7.4(↑0.4)	18.0(↑0.3)
	-20	-3.10	22	-23.3(↑1.2)	-48.6(↑11.1)	7.8(↑0.2)	21.2(↑0.2)
	-17	-3.10	22	-22.9(↑1.4)	-47.5(↑12.0)	11.7(↑0.4)	24.0(↑0.2)
	-14	-3.00	22	-23.5(↑1.7)	-48.5(↑12.0)	15.5(↑0.6)	26.4(↓0.4)
vs	-11	-3.05	16	-22.7(↑1.0)	-40.2(↑3.7)	27.3(↑1.7)	27.4(↑0.4)
Fixed	-23	-3.1	28	-23.1(↑1.5)	-51.2(↑15.4)	3.3(↓0.2)	18.4(↓0.1)
	-20			-23.1(↑2.0)	-48.9(↑14.9)	6.6(↓0.2)	22.2(↓0.1)
	-17			-22.9(↑3.0)	-46.3(↑14.8)	9.5(↓0.9)	25.0(↓0.4)
	-14			-22.9(↑4.4)	-43.3(↑14.6)	13.9(↓0.1)	28.1(↓0.1)
	-11			-22.5(↑6.0)	-38.6(↑12.9)	23.6(↑0.1)	31.9(0)

On the other hand, the results with fixed drain and gate voltage configurations cannot always reach the ACLR threshold, because the required input signal takes the PA into hard saturation. In addition, when the ACLR requirements can be met, the efficiency performance is lower than when considering the optimal configuration. For example, if we consider the first row of the fixed configuration and the second row of the optimal configuration (background coloured in brown in Table 3.8), they have similar output power, but the optimal configuration has two times the efficiency of the fixed configuration. Although the overall PA efficiency is not very high due to the limitation on available input power to the PA (due to the pre-PA gain), the experimental results shown have validated the dynamic supply strategy to maximize the PA efficiency when considering different output power levels. Moreover, DPD has been proved necessary to guarantee the required linearity levels specified in communications standards.

CHAPTER 4. FPGA IMPLEMENTATION

This chapter presents the efforts to implement the IFFT and DPD in a FPGA. The IFFT is essential for the OFDM signal generation, and the DPD plays a central role in guaranteeing the linearity of the output signal. Both of them were implemented by using the high-level synthesis (HLS) tool provided by Xilinx, and can be exported to intellectual property (IP) cores that enable synthesizing and implementing in the Vivado design suite. The IFFT has already provided by the HLS library, so the efforts were mostly put on analysing the documentation and using it correctly. However, the DPD is not provided in the HLS library and has to be designed carefully from scratch and make sure of the logical correctness. To optimise the resources usage and to meet the throughput requirements, it is also required to put proper directives in the HLS design.

In the following subchapters we will first introduce the FPGA device, explain the Zynq SoC chip architecture and the design principle. Then, we will discuss the HLS design methodology. Finally, we will provide the design of the IFFT and the DPD blocks.

4.1. FPGA and Zynq SoC introduction

A Field-Programmable Gate Array (FPGA) contains arrays of programmable logic blocks, and hierarchies of reconfigurable interconnect that allow blocks to be wired and perform any kind of digital logic operations. The highly flexible nature and the large amount of logic resources enable engineers to design hardware optimized implementation for algorithm. By pipelining the process or parallelizing the computation, it is especially suitable for those timing sensitive applications such as real-time video processing and communications systems.

Although the FPGA can be programmed to implement various digital components/functions (i.e. adder, multiplier, GPU, CPU, etc.), there are some trade-offs between performance, cost and power consumption. For example, the Xilinx Virtex-7 can be configured to Microblaze processor which is a soft core that can run in the maximum clock frequency of 343 MHz [30]. In comparison to the ARM Cortex-A9 series that can run in a maximum CPU clock of 2 GHz, the soft core has lower performance. Also, the soft core occupies a large area of logic resources and thus high-cost. For most complicated applications that want to keep the convenience of using the existing library from high-level languages, the central processing unit (CPU) or microprocessing unit (MPU) cannot be avoided. One solution is to make the FPGA and the high-performance MPU existing in a single circuit board to have the advantages of both. However, when referring to high data rates, the physical interfaces between the FPGA and the MPU have to be able to function at high frequencies. Hence, the board designer has to consider the high-speed path and its signal integrity. Once the circuit board is produced, modifications are unlikely to be made.

For these reasons, the market started offering board level products that equipped with FPGA and MPU, for example, the ARM MPS3 board [31]. With the high-speed interface already well designed, clients can select the desired board according to the specific needs in performance, size, and the power consumption. Some FPGA vendors started combining their product with the popular MPU architectures, for example, Xilinx marketed the Zynq as an All-Programmable SoC (APSoC), the architecture of which is shown in the following subchapter.

4.1.1. Zynq APSoC architecture

The Zynq is a device that combines a dual or single core ARM Cortex-A9 processor with traditional FPGA logic fabric. The functional block diagram of the Zynq-7000 family is shown in the figure below.

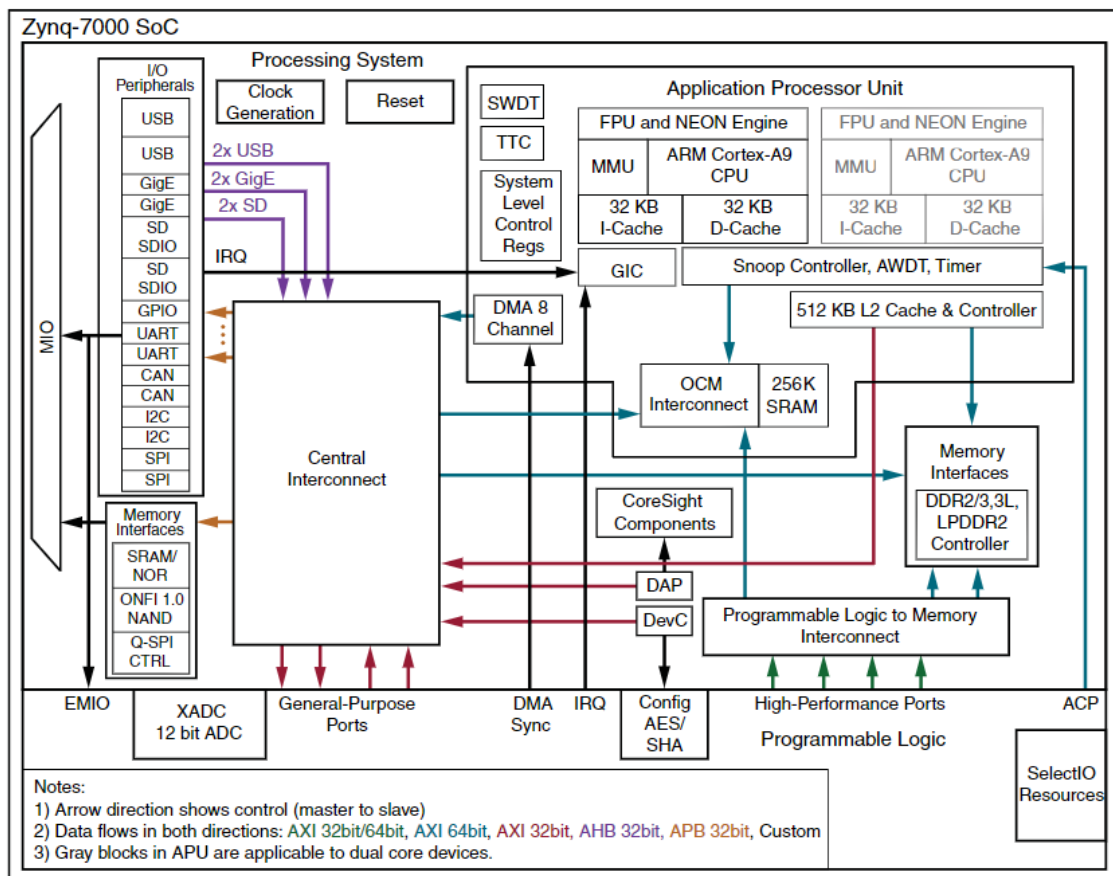


Fig. 4.1 Zynq-7000 SoC block diagram [32]

The Zynq architecture conveniently maps the programmable logic (PL) and the processing system (PS), i.e., the ARM part, respectively. The PS and PL are on separate power domains, enabling the user to power down the PL for power management. The PS part is composed of an application processor unit (APU), memory interfaces, I/O peripherals, and interconnection. Without considering the interconnection, the PS is just a traditional ARM processor. The interconnect interfaces are the infrastructures that enable communication between the PS and PL. In comparison to those board level FPGA and ARM solutions, it has shorter

current return path and guarantees better signal integrity, which are good properties of high-speed digital I/O. The interconnection provides several interface options to users, for example, the high-performance port that allows high-speed and high-throughput data transfer directly between the APU and the PL; and the general-purpose ports that ease the memory mapping port design, which is useful for users to design their extra/custom peripherals.

With the Zynq SoC, developers can reconsider their application, and make their implementation more profitable and efficient. The presence of the Zynq also promoted the Zynq-based application design methodology.

4.1.2. Zynq-based design methodology and practice

In this subchapter, the overview of the UltraFast embedded design methodology presented in [33] is introduced. By following the design methodology, the general consideration for implementing the proposed communications system for UAS applications are discussed.

The UltraFast design methodology consists of five parts: system level considerations, hardware design consideration, software design considerations, hardware design flow, software design flow and debug. Each part conveys key principles, specific to do and not to do and avoiding pitfalls, which are valuable information for a developer or team. The following table lists some system level, hardware and software design considerations and the factors that are related to the communications system of this thesis.

Table 4.1. Implementation considerations

System Level Considerations	
Performance	The overall system performance is the first thing to have in mind. The type of application decides the design goals. In this work, the drone communication system is required to process the payload data and stream it to the RF up-converter continuously.
Power Consumptions	Power consumption is an important factor for system architects and board designers. In this work, it is expected to use the automatic power supply and DPD to improve the power efficiency of the RF front-end, however, if the power consumption for the DPD core implemented in the FPGA is too high, then this system makes little sense. After the hardware implementation, it is necessary to pay attention to the power consumptions.
Clocking	The clocking resources are also important to understand, since the PL and PS can have many function blocks that run in different clock domains. Especially for the communication system, sampling rate is a crucial parameter that usually have a direct relationship to the driving clock. In this thesis, the

	OFDM was supposed to produce data at 20 MSa/s, and the DPD has to handle 60 MSa/s baseband data.
Interrupts	The system-level interrupt provides a comprehensive set of capabilities for a developer to prioritize hardware and software resources in the application. In the automatic TX power control part, the power supply needs to be configured when the desired TX power is changed. The dynamic configuration can be triggered by specifying an interrupt signal.
Partitioning	Partitioning is an important system-level decision to make. It is about how to partition the functions of an application between using the hardware resources (PL) and the software resources (PS). The planned system comprises an automatic TX power controller and an OFDM communication system with DPD. Apparently, the PL is naturally good at handling the data stream of communication system. But it is not a wise decision to use the PL to implement the Kalman filter demanded by the automatic power controller, because the data rate is comparably slow and the link margin allows to take action with little delay.
Hardware Design Considerations	
Memory Interfaces	DDR interface can be used to read/write the RAM and the QSPI interface can be used to access the flash memory.
Peripherals	The Zynq peripherals are memory-mapped and can be categorized as PS peripherals and PL peripherals. The PS part has provided many commonly used peripherals such as USB, CAN, UART, Ethernet, and SPI. For retrieving GPS data from the flight controller, the existing PS SPI interface can be used. But for those peripherals that are not common, for example, the interface to the DAC, it might require to custom a specific PL peripheral or import the IP core provided by the device vendor.
Designing IP Blocks	Some IP blocks are out-of-box and free to use which help reduce time-to-market. For example, later in this chapter, the IFFT IP is introduced. While other specific or custom algorithms like the DPD has to be implemented from scratch.
Hardware Performance	There are two important metrics of the PL performance: throughput and latency. Depending on the data path of the application, the hardware will be throughput constrained or latency constrained. In the RF communication system, throughput is always put in the first place. When the throughput is satisfied, more efforts can be devoted to decrease the latency.
Dataflow	The dataflow describes how the data is transferring within PS, PL or between PS and PL. Dataflow within PS is maintained by the central interconnect that connects the APU, DDR and the PS peripherals. Dataflow within PL can be arbitrary because of the infinite possible FPGA designs. The most important dataflow that a Zynq developer needs to consider

	is the interconnection between PS and PL. As shown in the Zynq block diagram in Fig. 4.1, several options are available, depending on the required hardware performance. For example, in this thesis, the low-speed automatic power controller can use the general-purpose port (GP) while the high-speed RF data path has to use the high-performance port (HP).
PL Clocking	The system clock requirement has already been taken into account at the system level. In PL clocking, we considered selecting the clock sources. The Zynq system has four clock sources: Clock from PS (FCLK), Clock from a GT (Gigabits Transmitter), Clock from external source and Clock generated by MMCM (Mixed-Mode Clock Manager). For the experiment setup just for validating, the FCLK was selected in this work.
Software Design Considerations	
OS and RTOS Choices	The embedded ARM-A9 is a powerful processor that can run bare metal software, RTOS or even general-purpose OS like Linux. Running a Linux system enables the use of a huge number of existing libraries for software development. While running the bare metal software has the advantages of small code footprint and guaranteed response time.
Libraries and Middleware	For running the bare metal software, Xilinx provides the SDK (software development kit) that has commonly used libraries including GNU C/C++, POSIX Pthread libraries, LWIP libraries, etc. For running the Linux OS, more options are available, for example, Python interpreter can be installed and libraries like the NumPy can also be used. It means that programs written in the theory validation stage can be used directly. For example, in the system-level considerations we decided that the automatic TX power controller logic would run in the PS part, so the only thing needed to be done was to trigger an interrupt and update the power supply settings.

In short, we have considered the following implementation approaches: partitioning the functions into PS and PL, pre-define reasonable data path and decide the application environment (whether to use an OS and selected libraries). Fig. 4.2 summarizes the implementation considerations that have been made for the proposed drone communication system.

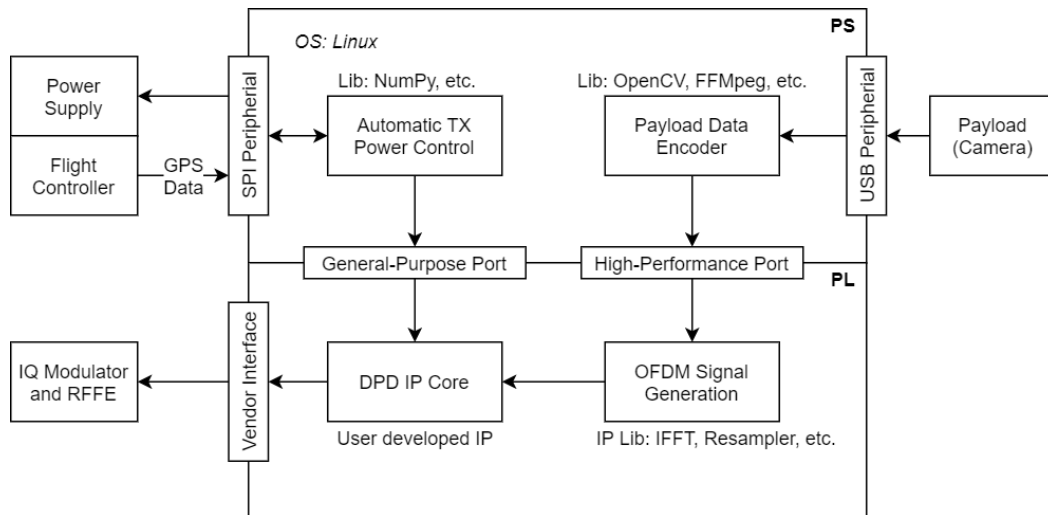


Fig. 4.2 Implementation considerations of the proposed drone communication system.

Apart from the three aforementioned considerations, the UltraFast methodology also includes the software and hardware design flow and the debug method. For hardware design, Xilinx provides the Vivado IDE, which is a design suite that can be used to build IP subsystems. Users can use Vivado to perform traditional register transfer level (RTL) design, or generate block design. The capability of creating and packaging IP for reuse significantly boosted the speed of development. Besides, Xilinx also provided the HLS tool for hardware IP design that can transform the high-level C/C++ design to RTL representation, which can be synthesized in the Vivado IDE. For software design, Xilinx SDK provides bare-metal drivers and build-in functions to generate boot support project such as the FSBL and the U-Boot required by the Linux boot process. Both Vivado and Xilinx SDK include supports for debugging the hardware design and the software design, respectively. Fig. 4.3 shows the complete chain of hardware and software development flow.

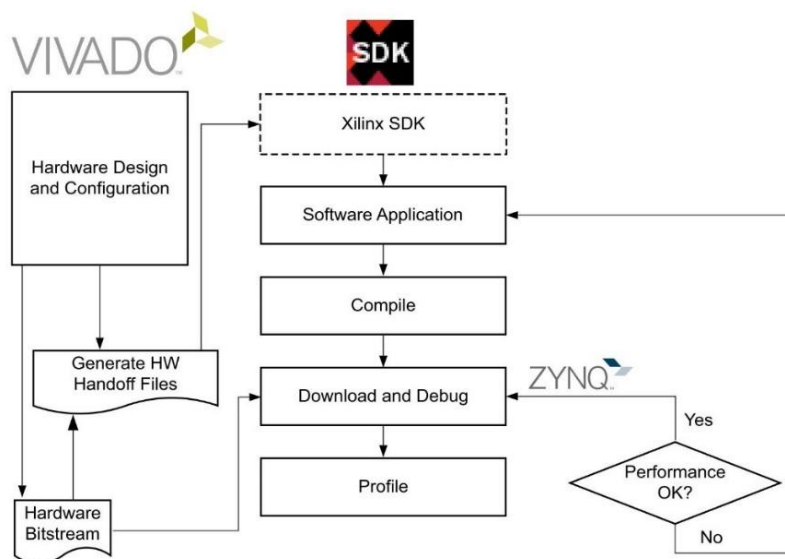


Fig. 4.3 Hardware and software development flow [33]

4.1.3. Useful PL resources

To have the best practices in the design of Zynq-based application, it is also important for developers to be aware of the available on-chip PL resources. Typically, a general-purpose FPGA is composed of logic fabric, which comprised slices, configurable logic blocks (CLB) and input/output blocks (IOB) for interfacing. In the Zynq PL, each CLB contains two logic slices. Within a slice, there are some lookup tables (LUT) and flip-flop (FF) resources. Next to each CLB, a switch matrix is used to provide flexible routing facility for making connections between elements within a CLB or other resources on the PL. The Zynq logic fabric and its constituent elements is shown in Fig. 4.4.

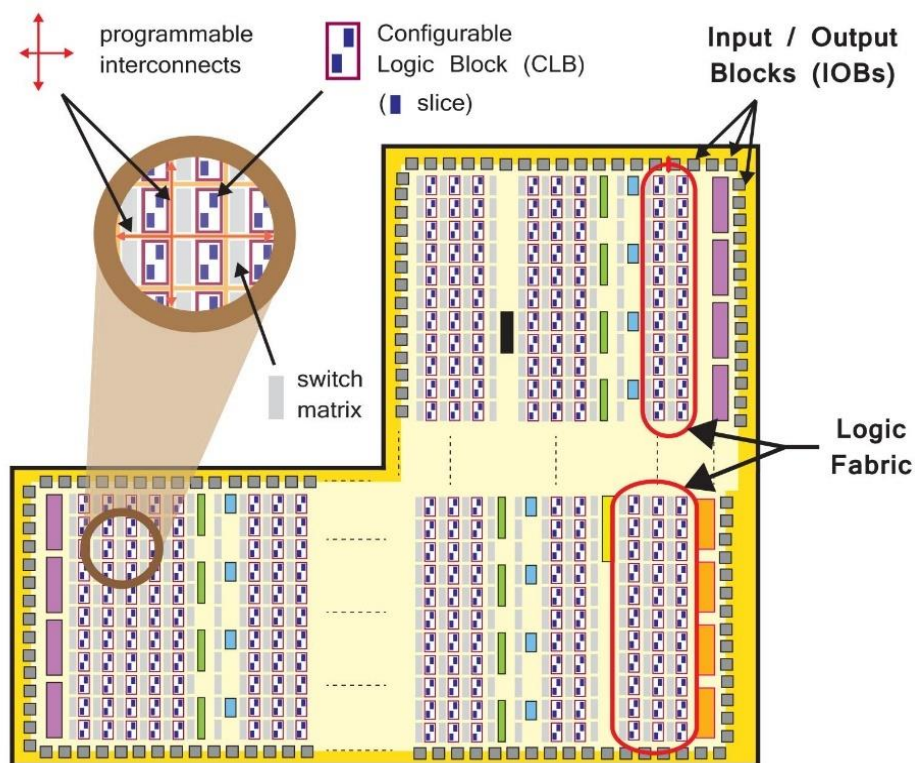


Fig. 4.4 The logic fabric and its constituent elements [34]

In addition to the general fabric resources that every FPGA should have, the Zynq PL also provides special purpose resources: the DSP48E1 and the Block RAM. For the implementation of the DPD linearizer in this thesis, DSP48E1s were used for high-speed arithmetic, while to save logic fabric resources, the Block RAMs were used for storing the look-up tables (LUTs) containing the complex gains, which are frequently accessed by the DPD.

DSP48E1

DSP applications conventionally use many multipliers and accumulators, which can be implemented with the FPGA logic slices or by using the dedicated DSP

slices. DSP48E1 slices are the DSP elements available in Xilinx's 7 series FPGAs, which provide improved flexibility and utilization, improved efficiency of applications, reduced overall power consumption, and increased maximum frequency [35].

A basic DSP48E1 slice supports 25-bit x 18-bit two's-complement multiplier with dynamic bypass functionality, 48-bit accumulator that can be used as a synchronous up/down counter, power saving pre-adder and pipelining capability for cascading. It has a single-instruction-multiple-data (SIMD) arithmetic unit that can handle add/subtract/accumulate of dual 24-bit or quad 12-bit input data at one instruction period. The DSP48E1 slice can handle ten different logic functions of the two operands, controlled by the optional logic unit. The simplified functionality of the DSP48E1 slice is shown in Fig. 4.5.

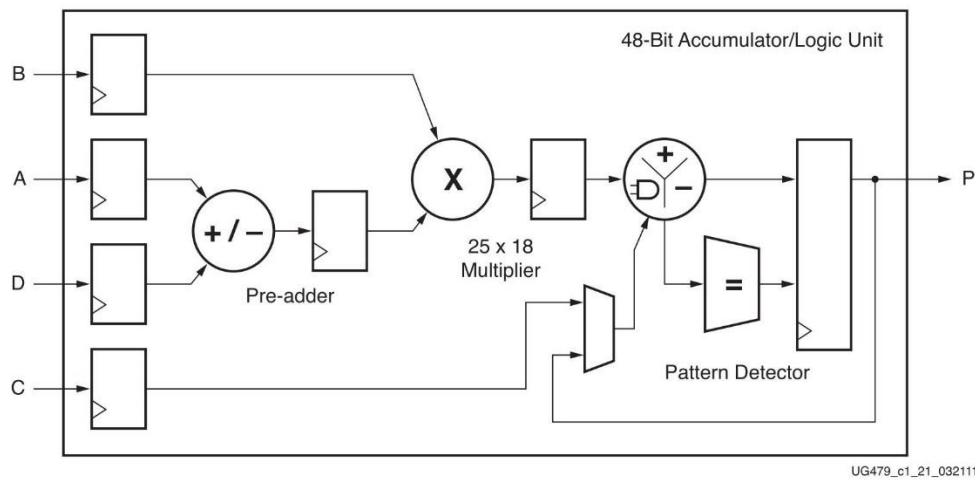


Fig. 4.5 Simplified DSP48E1 slice functionality [35]

From the slice functionality, it can be clearly seen that the DSP48E1 is good at handling operations such as the ones described in (4,1). These operations are frequently used in the DSP application such as FIR filter but they are also useful for the DPD implementation. The equation of this simplified DPS48E1 is shown below, where P' represents the former result.

$$P = P' \pm B \times (A \pm B) \quad (4,1)$$

Block RAM

The Block RAMs in the Zynq can implement Random Access Memory (RAM), Read Only Memory (ROM), and First In First Out (FIFO) buffers. A large amount of data can be stored in a small physical space on the device by using the Block RAM resources, offering high-speed accessing capability. Each Block RAM can store up to 36 Kb of information and are independent to read or write. In comparison to the DDR RAM, parallel accessibility significantly improves the loading speed. In comparison to the Distributed RAM solution, which is constructed from the LUTs resources within the logic fabric, the Block RAM solution can save a large area of CLBs.

Each 36 Kb block RAM can be configured as one 36 Kb RAM (SDP mode) or two independent 18 Kb RAMs (TDP mode) with their own access port. Each port has its own address, data in, data out, clock, and write enable. Each 18 Kb RAM has a 16-bit address bus and a 36-bit data bus. It is not mandatory to use the full 36-bit, instead, the actual data width can also be configured to 1, 2, 4, 9 or 18-bit according to the needs [36]. Similarly, the address bus can also be partitioned to increase the density for storing data with small width. For example, in TDP mode, the Block RAM has a configuration of 18-bit data width and 10-bit address, which results a 1024 depth memory pool.


In the DPD IP core implementation, the Block RAM resources were used for the LUT-based implementation of the DPD function, i.e., for storing the gains or coefficients that were frequently accessed by the computation. The address bus and data bus specifications will be conditioned by the fixed-point data width of the LUT.

4.1.4. The Zedboard for experimental validation

The evaluation of the FPGA implementation was carried out in the Zedboard [37]. It is a low-cost development board for the Xilinx Zynq-7000 APSoC that contains everything necessary to create a Linux, Android or other RTOS-based design. It also includes the Xilinx Vivado WebPACK support that enable users to evaluate all the Vivado features without extra charge, including the HLS tool.

The bench mark of resource usage of the FPGA design (shown later in this thesis) are all based on the Zedboard resources (the specific Zynq processor). For example, 50% of the LUTs represents half of the LUT resources inside the Zynq chip that were used for implementing the design. The key features of the Zedboard are described in Table 4.2.

Table 4.2. Key features of the Zedboard

	Processor: Zynq-7000 APSoC XC7Z020-CLG484-1	
	Programmable Logic Cells	85K
	Look-Up Tables (LUTs)	53,200
	Flip-Flops	106,400
	Block RAM (#36Kb Blocks)	4.9 Mb (140)
	DSP Slices (18x25 MACCs)	220
Memory	512 MB DDR3, 256 Mb Quad-SPI Flash, SD card support	
Interface	USB-JTAG, 10/100/1000M Ethernet	
Connectors	FMC-LPC connector, 5 Pmod header, Agile Mixed Signaling	
Clocking	33.33333 MHz clock for PS, 100 MHz oscillator for PL	
Display Port	HDMI, VGA	
GPIO	8 LEDs, 7 push buttons, 8 DIP switches	

4.2. High-Level synthesis

Unlike the development of MPU or CPU applications that can use the high-level programming languages such as C, C++ and Python, typically, FPGA designs require the use of hardware description languages (HDL) such as VHDL and Verilog HDL. This required experienced hardware engineers since it was not friendly for novices. In recent past years, companies like Xilinx and Intel have released their high-level synthesis (HLS) solutions (e.g., Vivado HLS, Intel HLS Compiler for Quartus) that ease the hardware design by letting developers to use high-level languages such as C, C++ and System C.

The general HLS includes three phases to convert the C code into an RTL implementation: Scheduling, Binding, and Control logic extraction.

- **Scheduling** determines which operations occur during each clock cycle based on the length of the clock cycle, and the time it takes for operation to complete (defined by the target device hardware specs). If the clock period is longer than the operations, they will be automatically scheduled within one clock cycle by the HLS. Otherwise, the HLS will schedule these operations over more clock cycles.
- **Binding** determines which hardware resource to use for each scheduled operation. The final performance may be affected by using different resources, for example, using DSP48E1 for multiply operation is more efficient than using the logic slices.
- **Control logic extraction** creates a finite state machine (FSM) that sequences the operations in the RTL design. It guarantees the logical correctness and makes the function work as expected.

The developer can make decisions on the above phases by specifying directives. Directives are pragmas in the C code that are used to inform the HLS compiler. The Vivado HLS provides optimization directives to constraint the input/output port, perform pipelining, loop unrolling, and specify resources. The Vivado HLS C++ library provides many useful math functions that are already optimized by directives. In addition, the HLS library provides arbitrary precision data types support for both RTL implementation and C-Simulation [38]. In the following subchapters, useful HLS features are described.

4.2.1. Fixed-point support

The Vivado HLS C/C++ library supports arbitrary precision data types for both integer and fixed-point data types. The advantages of using arbitrary precision data types is that it allows the hardware to use a small width data bus which saves the PL resources. Converting floating point to fixed point can also lower the power consumption and improve the latency of DSP calculation [39]. However, the fixed-point type has lower dynamic range and fewer precision than the floating-point data type. When using the fixed-point type, the designer has to decide the precision and the position of decimal point to meet the minimal requirement of numerical precision. Fortunately, the HLS tool can emulate the functions with different precision setup, which helps the developer to fast validate and make a

proper decision. Furthermore, the developer should determine the behavior of overflowing and quantization to avoid outliers when the calculation excess the boundary. Table 4.3 summarizes the behavior of different overflow modes and quantization modes that support by the HLS.

Table 4.3. Overflow and quantization modes

Overflow Mode	Description
AP_SAT	When overflow, saturate to max/min value
AP_SAT_ZERO	Set to 0 when saturation happen
AP_SAT_SYM	Symmetrical saturation
AP_WRAP	Wrap around the dynamic range
AP_WRAP_SM	Sign magnitude wrap around
Quantization Mode	
AP_RND	Round to plus infinity
AP_RND_ZERO	Round to zero
AP_RND_MIN_INF	Round to minus infinity
AP_RND_INF	Round to infinity
AP_RND_CONV	Convergent rounding
AP_TRN	Truncation to minus infinity
AP_TRN_ZERO	Truncation to zero

The description of the quantization mode might be confusing, to help better understand the different quantization modes, Fig. 4.6 , taken from [40], gives a direct view of the quantization behavior.

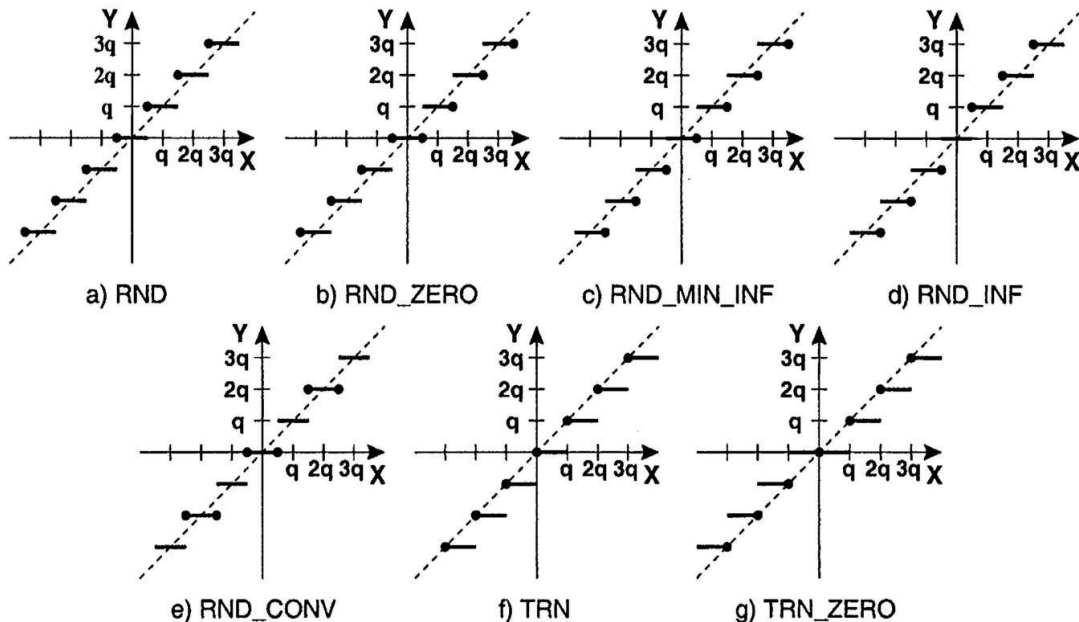


Fig. 4.6 Different quantization modes [40]

4.2.2. Directives for optimization

The Vivado HLS provides up to 24 optimization directives. The user can apply different directives to the design according to the requirements on throughput, latency, area, etc. In this subchapter, two directives that have been used in the DPD implementation for increasing the throughput are presented: pipeline and array partition.

The **pipeline** directive can be stated to a function or a loop to allow operations to happen concurrently, so that at each execution step the next operation can begin without waiting for all the operations to finish. One important metric for the pipelining is the initiation interval (II), which implies how many clock cycles are required to start the next iteration. For most DSP applications, such as the DPD, a new input sample is read every clock cycle (II=1). The initiation interval is a parameter of the pipeline directive, when specified a number, the HLS will try to schedule the RTL to satisfy the throughput. If it fails to satisfy it, the HLS will increase the II by one and try again until the design can be synthesised. Several reasons may restrict the design from reaching the II of one, such as the data dependencies or memory dependencies between the different iterations of a loop. The memory port limitation can also lead to a throughput bottleneck since, as discussed in the Block RAM subchapter, an 18 Kb Block RAM only has one access port. If an array is implemented using one Block RAM resource, and one operation needs to read multiple elements in that array, then it has to be scheduled in several clock cycles. To resolve the memory port limitation, the array partition directive can be used.

The **array partition** directive can split the array into multiple memory blocks to increase the number of ports, and improve the throughput of a read/write intensive algorithm. The HLS provides three types of array partitioning: block, cyclic, and complete. The block type splits the array into equally sized blocks of consecutive elements; The cyclic type splits the origin array into equally sized blocks of interleaving elements; The complete type completely split the array into individual elements. Fig. 4.7 shows the behavior of different array partitioning types.

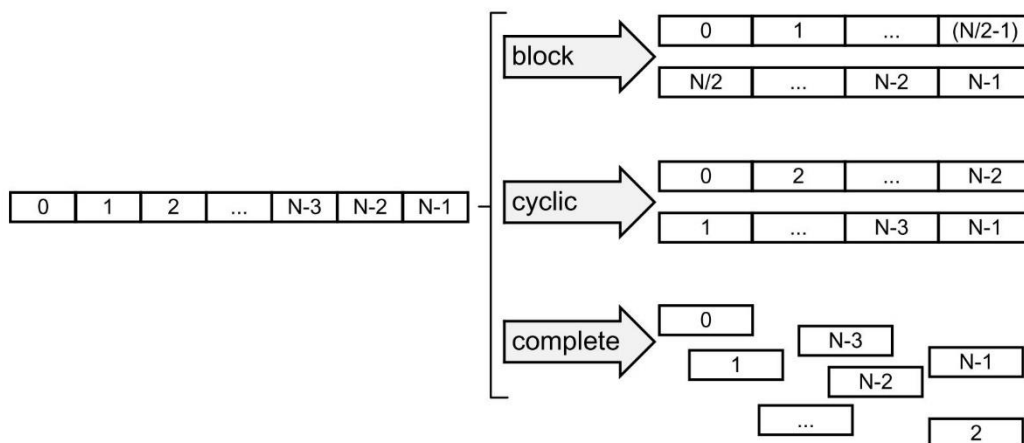


Fig. 4.7 Different types of array partitioning [38]

4.2.3. Generating in-chip interfaces

As mentioned before, the Zynq APSoC comprises the PS and the PL, which are connected by the interconnection. In the system consideration, we decided that the PS is responsible for encoding the payload data, and then stream it to the PL by the high-performance port. In addition, the automatic power control logic is deployed in the PS and the control commands are sent by the general-purpose port. To establish these connections, the PL logic blocks must come with the proper interfaces.

The Vivado HLS can generate I/O interface wrapper for the IP block automatically. Developers can specify which kind of interface to generate by applying interface directives to the I/O variables of the top function. The Vivado HLS supports several I/O protocols based on efficient industry standard interfaces such as the AXI4, which are interfaces protocol created by ARM.

The Advanced eXtensible Interface (AXI) is part of the ARM Advanced Microcontroller Bus Architecture (AMBA), and supports high-bandwidth and low-latency designs without using complex bridges. Also, it is suitable for memory controllers with high initial access latency, and provides flexibility in the implementation of interconnected architectures. The AXI4 is the 4th generation of the AXI protocol, including the definition of AXI4, AXI4-Lite [41] and AXI4-Stream [42] protocol. AXI4 and AXI4-Lite are memory mapped master-slave protocol, transactions are initialized by the master and the read/write operations are performed to address. AXI4-Stream is also a master-slave protocol, but is not memory mapped, it is designed for high-throughput master to slave data transfer. The Vivado HLS supports the port generation of AXI4-Lite and AXI4-Stream, which saves the developer from reminding the details of the AXI4 specifications. In this work, the AXI4-Lite is used as the general-purpose port for controlling the TX power. While AXI4-Stream is specified for streaming the encoded payload data and the RF baseband data.

4.3. IFFT implementation

Inverse discrete Fourier transform (IDFT) was used in the OFDM signal generation. It was implemented by using the computationally efficient inverse fast Fourier transform (IFFT), which is still a calculation intensive algorithm. FPGA is good at handling the IFFT algorithm by using optimized hardware implementation architectures. The IFFT is popular and commonly used in the field of DSP, and is provided by the HLS library.

4.3.1. FFT IP from the HLS library

The IFFT is included in the FFT IP library and specified by the IP configuration. In fact, the FFT in the HLS library is a high-level wrapper of the Xilinx LogiCORE

FFT IP [43] product. To use the FFT library, it is essential to read the IP specification and learn about its usage, performance and limitation.

The FFT/IFFT core computes N-point DFT or IDFT where $N = 2^m$ and m is in the range 3~16. It supports both fixed-point and single-precision floating-point data types. The IP core provides four architecture options to offer a trade-off between size and throughput: Pipeline Streaming I/O, Radix-4 Burst I/O, Radix-2 Burst I/O and Radix-2 Lite Burst I/O. The IFFT is computed by conjugating the phase factors of the corresponding forward FFT. However, the $\frac{1}{N}$ scaling is not implemented in the IFFT, which is tricky and should be kept in mind. The equation of the IFFT IP (without the scaling) is given by

$$x[n] = \sum_{k=0}^{N-1} X[k]e^{j2\pi kn/N} \quad (4,2)$$

4.3.2. Evaluate the IFFT IP

As described in the OFDM design subchapter, a 64-IFFT was used to generate the 20 MHz bandwidth baseband complex data. So, the design goal of the IFFT block is to meet the 20 MHz bandwidth requirement. The clock for the IP block was set to 100 MHz, the fixed-point data type in width of 16 was used, and the input data was fed by batches every 64 samples. The HLS synthesized report with different architecture settings is shown in Table 4.4.

Table 4.4. IFFT performance and resources of different architecture

Performance / Resources Usage	Pipeline Streaming	Radix-4 Burst	Radix-2 Burst	Radix-2 Lite Burst
Latency	281	245	423	549
Initiation Interval	282	246	424	550
Equivalent Bandwidth	22.7 MSa/s	26.0 MSa/s	15.1 MSa/s	11.6 MSa/s
MSE (vs float type)	-88.9 dB	-88.9 dB	-88.9 dB	-88.9 dB
BRAME_18K	6 (2%)	11 (3%)	7 (2%)	6 (2%)
DSP48E	6 (2%)	9 (4%)	3 (~0%)	2 (1%)
Flip Flop	5495 (5%)	5495 (5%)	5495 (5%)	5495 (5%)
Look-Up Table	4499 (8%)	4499 (8%)	4490 (8%)	4490 (8%)

Table 4.4 shows that the Pipeline Streaming and the Radix-4 Burst architecture are fast enough to meet the 20 MHz bandwidth, and the resources usage of the Pipeline Streaming is slightly lower than the Radix-4 Burst. Therefore, the IFFT IP can be implemented in the Zynq and satisfy the OFDM application with an acceptable resource usage.

4.4. DPD implementation

In this subchapter, the FPGA implementation of DPD will be presented. It will be implemented as a reusable IP block achieved by writing custom C++ HLS codes. First, the FPGA architectures of DPD will be presented and discussed. Then a universal LUT-based DPD architecture for memory-polynomial-based models will be proposed. Finally, the proposed architecture will be synthesised and tested with the experimental setup in the lab, and its performance and resources usage will be analysed.

4.4.1. Architectures of implementing DPD in FPGA

The DPD algorithm can be deployed in the FPGA either using the polynomial calculation-based method or the LUT-based method. The calculation-based method is to fork the behaviour model in the PL and perform exactly the same calculation as the polynomial-based model definition. In this way, lots of calculation resources are required, and the developer has to implement each model independently. The advantage is that with the polynomial calculation model is possible to achieve results as precise as in the high-level environment such as Matlab and Python, and it can immediately use the same coefficients. The LUT-based method does not require implementing the polynomial nonlinear functions, instead, it uses LUTs to recall the nonlinear behaviour. When new input data comes in, it indexes the LUTs and calculates the output with the restored values. In this way, the calculation time and resources are significantly saved, which helps to lower the latency and increasing the throughput. The price to pay is that needs more memory resources for implementing the LUTs.

There are some works that have implemented the LUT-based DPD. For example, in [44] the authors propose the combination of basic predistortion cells (BPCs) to compensate not only for the nonlinear behaviour but also for the PA memory effects. A low-cost implementation of the dynamic deviation reduction-based (DDR) Volterra model was proposed in [45], which takes into account the lagging cross-memory terms. A PLUME model for FPGA implementation was developed in [46] and compared with a GMP implementation that takes into account the nearest leading cross-memory items. All these works follow the same pattern described in [44], that the DPD model can be described combining BPCs, which required the use of LUTs. By extending the BPC architecture, the GMP model in (3,33) can be described following a LUT-based approach as

$$\begin{aligned}
 \hat{y}(n) = & \sum_{l=0}^{L_a-1} x(n-l)g_l(|x(n-l)|) \\
 & + \sum_{l_1=0}^{L_b-1} \sum_{m_1=1}^{M_b} x(n-l_1)g_{l_1,m_1}(|x(n-l_1-m_1)|) \\
 & + \sum_{l_2=0}^{L_c-1} \sum_{m_2=1}^{M_c} x(n-l_2)g_{l_2,m_2}(|x(n-l_2+m_2)|)
 \end{aligned} \tag{4,3}$$

where the $g_{i,j}$ represents the BPC (containing a LUT). For the GMP model, the $g_{i,j}$ includes all the correspond power terms and its coefficients. This can also work for the piecewise spline-based DPD [47] by replacing the polynomial kernels with the B-Spline kernel. The coverage of the cross-memory terms can be selected by specifying the parameters $(L_a, L_b, L_c, M_b, M_c)$. The block diagram of the proposed LUT-based DPD implementation of the GMP model on the FPGA is shown in Fig. 4.8.

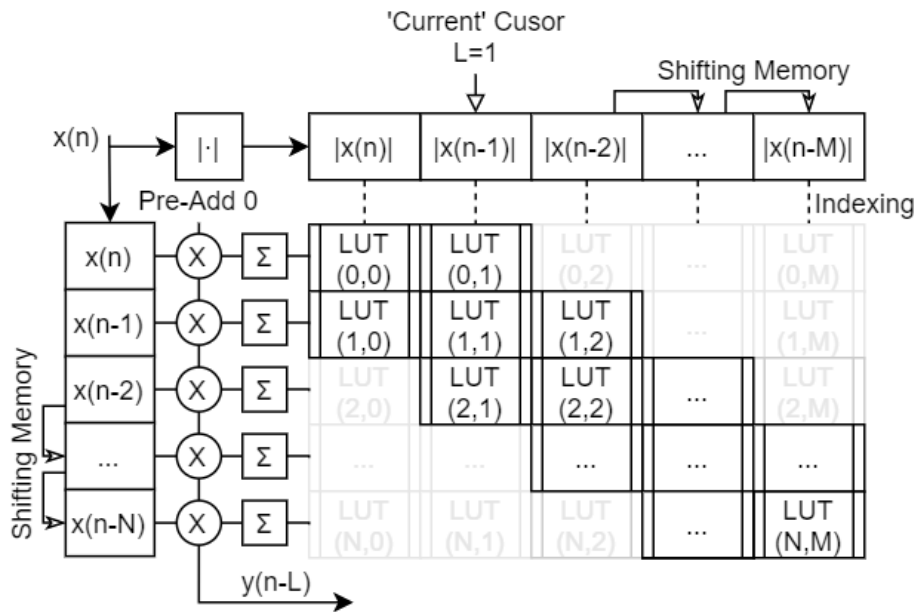


Fig. 4.8 Block diagram of the proposed LUT-based DPD implementation

The 2-D LUT distribution can represent any possible combination of cross-memory terms. Some of the LUTs can be disabled if they are not used and thus they will not be synthesized. Fig. 4.8 shows a GMP behavioural model implemented with LUTs including the nearest (delay is one) leading and lagging cross terms. The unused LUTs are coloured in grey. The 'Current' Cursor in Fig. 4.8 pointing to the shifting memory at the position of $(n - L)$, where $L = M_c$ is the number of leading terms in (4,3), represents the delay of the system.

The LUTs are indexed with the absolute value of the input (and past samples of the input). The absolute value operation is calculated using the CORDIC algorithm (provided by the HLS library). In each row, the retrieved LUT values (complex gains) are accumulated before carrying out the multiplication. This way the number of operations is significantly reduced (note that a complex-multiplication operation needs two real-multiplication operation). As mentioned before, the DSP48E1 resources are good at calculating the multiplication. Apart from the already described elements, the proposed implementation includes pre-adders and supports the pipeline directive, which significantly increases the throughput.

The following procedures were developed to generate the DPD IP core: 1) Extract the LUTs from DPD model, 2) Build the indexing logic and simulate, 3) Synthesize and optimise the design, 4) Layout (implement) in FPGA.

4.4.2. LUTs extraction

The extraction step generates look-up tables from the DPD model. Each LUT belongs to a BPC whose output is the sum of the model kernel (i.e., for GMP is the sum of power terms). For simplicity, we have considered that every LUT will have the same size, and the size has to be properly determined. As shown in Fig. 4.9 small LUT sizes degrade the output precision and the DPD linearization performance, while a large LUT size requires a huge amount of memory resources.

In this work, the GMP model was built up with Python scripts following the object-oriented programming (OOP) paradigm. By adding some support functions to the GMP class, it can extract the LUTs to build the LUT-based GMP DPD in HLS. In addition, the Python can also simulate the LUT-based DPD with the extracted LUTs. It is a fast and efficient method for evaluating different settings, such as the LUT size. By using the Pluto-SDR PA testbed introduced in subchapter 3.3.4, Fig. 4.9 shows the power spectral density (PSD) without DPD and with DPD considering different LUT sizes, in particular: 128, 256, 512, and 1024. The testing configuration used the same configuration described in the fifth row of Table 3.8.

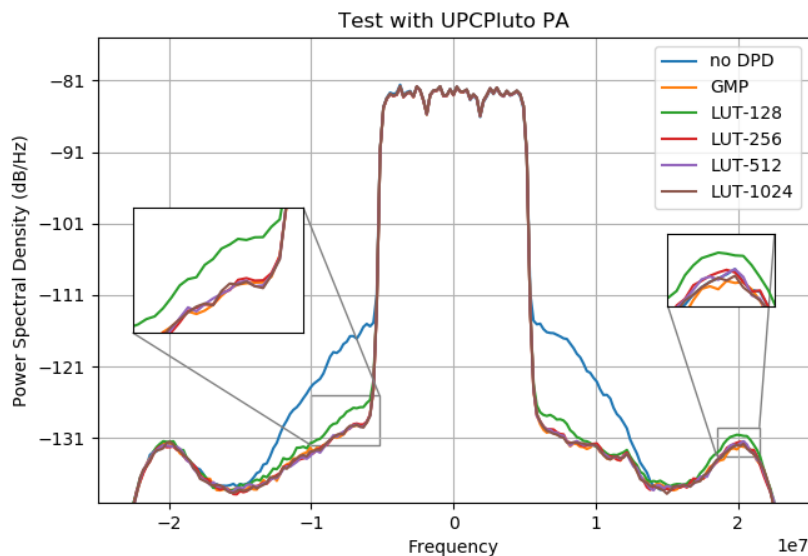


Fig. 4.9 DPD with different LUT size settings

Taking into account the trade-off between memory resources usage and DPD linearization accuracy, as shown in Fig. 4.9, a LUT size of 256 is good enough to reach the same performance as the original polynomial-based GMP model.

4.4.3. Indexing logic and simulation

As shown in Fig. 4.8, not all the LUTs were enabled, the HLS codes should only count for those useful cross memory part of the LUTs. For tradition C/C++ programming, it can be achieved by two nested for-loops and one if-condition sentence. However, in HLS, these for-loops and if-conditions will cause the implementation of extra logic in hardware, which will lead to unnecessary resources usage and variable dependencies that restricts the pipeline capability. The solution is to write these LUT indexing and math operation sentences line by line, which is a mission impossible for human if the LUTs are large-scale. In this work, an automatic HLS code generation programme based on the Jinja2 [48] templates engine was developed to manage this tough work.

The developed Python program will first analyse the DPD model, figure out how many LUTs are required and build a LUT index. Then it will infer a map of relationship between the cross memory, the corresponded LUT index, and the row index. Elements in the same row should have been summed up before the multiplication. Finally, the program will generate C-style HLS code. Fig. 4.10 shows the execution flow of the automatic HLS DPD code generation.

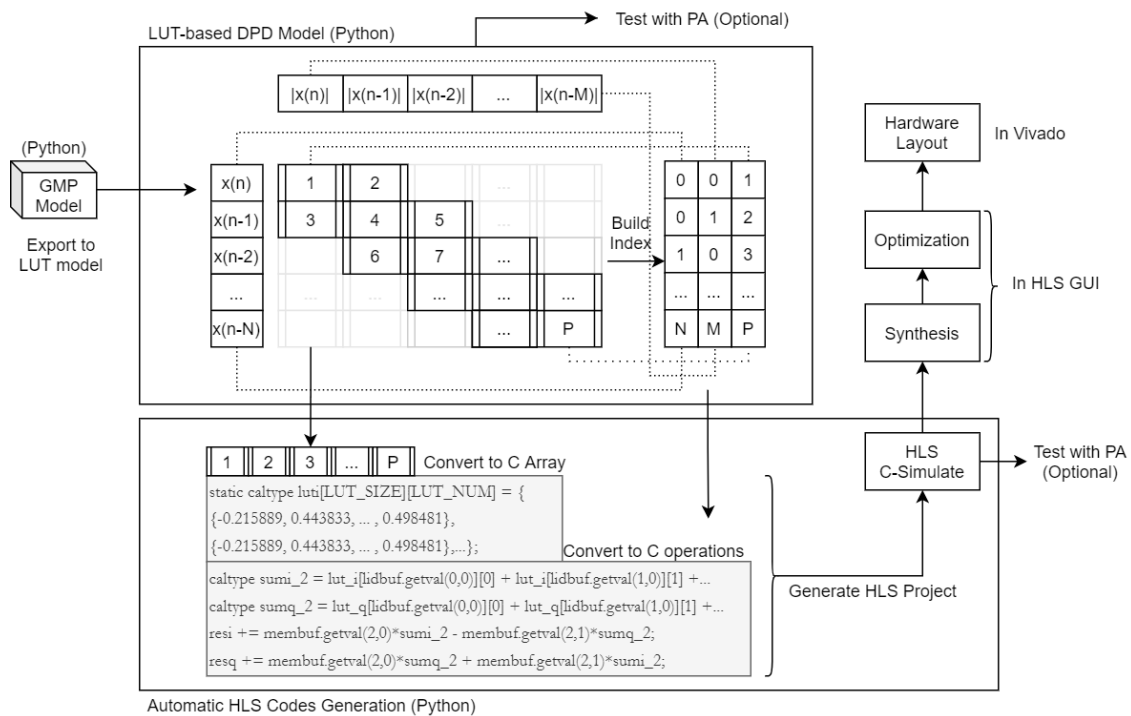


Fig. 4.10 Automatic HLS DPD code generation flow

The proposed automatic work flow increases the speed of implementing the DPD in FPGA. The high consistency of the results in Python, HLS C-simulation, and the final implementation in FPGA makes it possible to do a fast evaluation of the LUT-DPD performance. Fig. 4.11 and Table 4.5 show the comparison of the resulting PA output signal when considering the original GMP model described in equation (3,33), and the LUT-based model described in equation (4,3) . The LUT-based model implemented with Python (double data type) and the HLS C-

Simulation (fixed data type) are also compared. As mentioned before, the LUT size is set to 256.

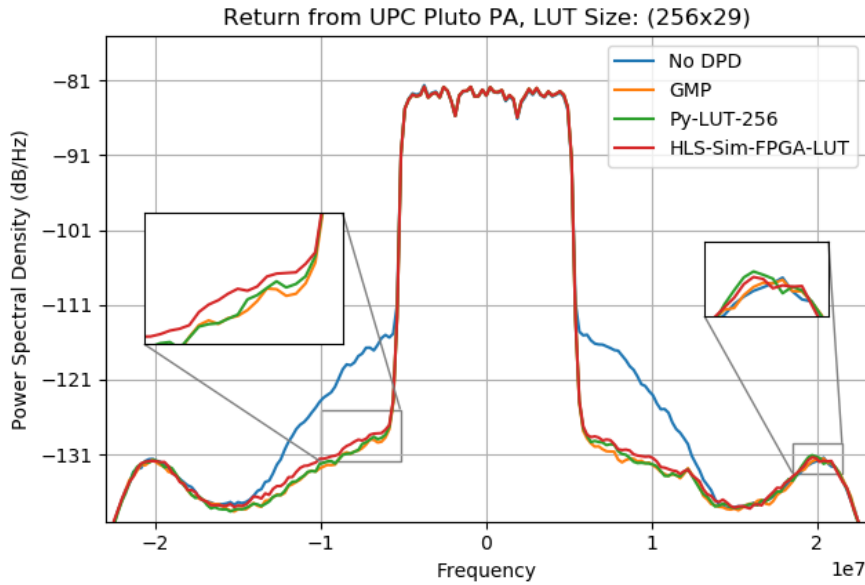


Fig. 4.11 DPD Performance GMP vs Py-LUT vs HLS-LUT

Table 4.5. DPD performance comparison

Type	NMSE (dB)	ACLR (dB)
GMP Origin	-23.45	-48.89
Python LUT (size 256)	-23.24	-48.56
HLS-C-Sim-FPGA	-22.89	-48.00

The result shows that the LUT-based DPD model has a small but acceptable performance degradation. At this point, the numerical part of the design has been validated and can be pushed to the synthesis stage.

4.4.4. Synthesis and performance analysis

As mentioned before, directives are essential in the HLS project to optimise the use of resources and guarantee the throughput. In this work, the pipeline directive was specified to the top function to allow concurrent execution of the DPD. Besides, array partition directive was specified to the LUTs array to give more memory ports so that the reading can take place in parallel. The OFDM signal was up-sampled by a factor of 3, which means that the DPD core should have at least 60 MSa/s of sample rate, hence the clock period of the IP was set to 16.67 ns. The synthesised design report is shown in the following table.

Table 4.6. Synthesised DPD performance and resources usage

Performance and Resources Usage	LUT Size 256x29	LUT Size 1024x29	LUT Size 2048x29
Latency	33	33	33
Initiation Interval (II)	1	1	1
Equivalent Sample Rate	60 MSa/s	60 MSa/s	60 MSa/s
BRAME_18K	58 (20%)	58 (20%)	113 (40%)
DSP48E	46 (20%)	46 (20%)	46 (20%)
Flip Flop	5316 (4%)	5320 (5%)	5322 (5%)
Look-Up Table	15146 (28%)	15156 (28%)	15161 (28%)

The table shows that the LUT-based DPD operating at the clock rate of 60 MHz can be successfully synthesised and meet the sample rate of 60 Msps (II=1). The resources usage of LUT size in 256 and in 1024 are almost the same, because the array partition directive forced to use individual Block RAM resources for each LUT column. When the LUT size was set to 2048, it doubled the Block RAM to store more LUT values. It means that for this specific design, LUT size of 1024 is more recommended.

Usually, the running condition of a PA is fixed and the LUT can be read only. However, in this work, the automatic TX power controller changes the power supply according to the position of the UAS, the DPD IP core should be able to adapt its coefficients. The DPD coefficients are estimated in Python (eventually can be carried out in the PS part) and then used to be derived into LUTs. It is possible to update the LUT contents in the Block RAMs by the PS through the AXI4-Lite interface. But the price to pay is the use of twice as much BRAM than the required in the Table 4.6. This is due to the way that the HLS synthesis the AXI4-Lite interface using double-buffering technique to allow read/write operations while the block is active.

4.4.5. Final implementation

In this subchapter presents the system-level integration of the generated DPD IP blocks in the Vivado IDE. System-level considerations have been made in the previous subchapter regarding the use of the high-performance interconnection for streaming high-speed baseband data between PS and PL. To be more specific, the HP port of the PS is set to enable, and the PS data stored in the DDR RAM is moved actively by the DMA controller. Then a memory map to stream (MM2S) converter is responsible to interface the AXI-Stream. For demonstration purpose only, Fig. 4.12 shows the system-level connection of the generated DPD core with the PS, both input and output ports were connected to the PS part. In a real application, the input of the DPD block should connect the output of the OFDM function block and the output of the DPD should connect the up-converter DAC driver block.

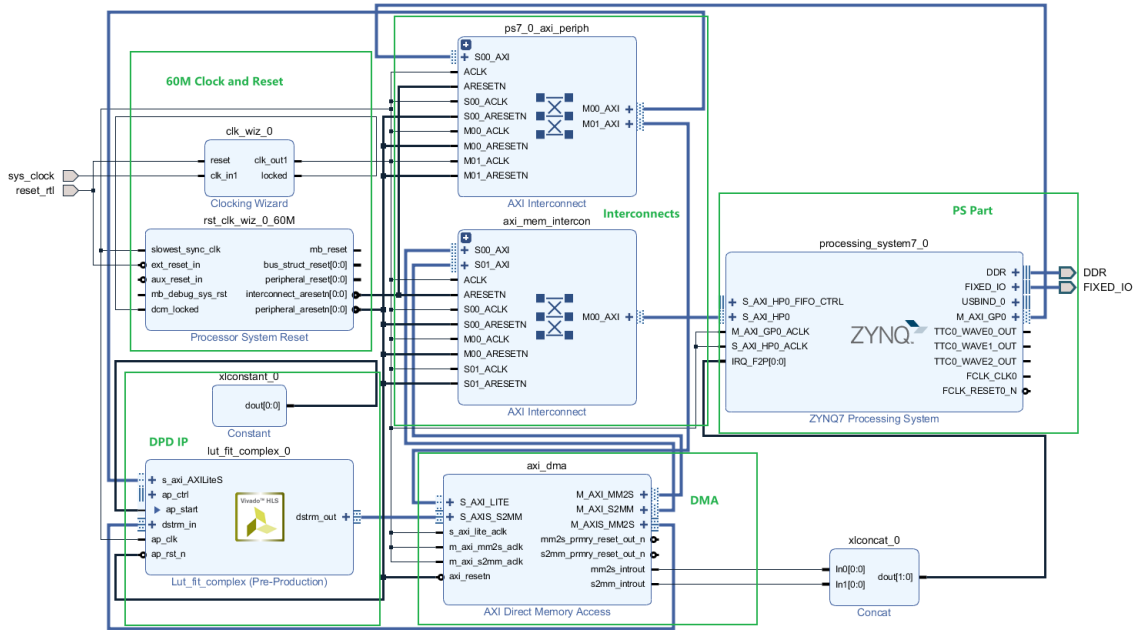


Fig. 4.12 System-level block design

The PS program can be developed in the Xilinx SDK IDE. If we are running Linux OS, it is also possible to use the cross-compiling environment. The driver logic is the same independently on where it is running (OS or bare-metal). For pushing/pulling data to/from the AXI-Stream interface of the DPD block, the PS will configure the AXI DMA block to setup the beginning of the DDR memory address and the data length, and initiate a transaction. Then the AXI DMA block will automatically move the data to the destination and raise an interrupt when the transaction is finished. To update the coefficients in the DPD block, the PS can directly set the new values to the corresponded BRAM by accessing the system memory according to the address offset, because the AXI-Lite interface has been memory-mapped to the system by a memory management unit (MMU). The address offset was generated automatically by the Vivado during the blocking design, it can also be manually specified through the address editor. Fig. 4.13 shows the address assignment of this design. Below the processing system, the 'lut_fit_complex_0' is the DPD block and the 'axi_dma' is the DMA block, both using the AXI-Lite interface. The AXI DMA manages the memory-map to stream (MM2S) and the stream to memory-map (S2MM), which was bound with the high-performance (HP) AXI port of the PS.

Cell	Slave Interface	Base Name	Offset Address	Range	High Address
processing_system7_0					
Data (32 address bits : 0x40000000 [1G])					
lut_fit_complex_0	s_axi_AXILiteS	Reg	0x43C0_0000	64K	0x43C0_FFFF
axi_dma	S_AXI_LITE	Reg	0x4040_0000	64K	0x4040_FFFF
axi_dma					
Data_MM2S (32 address bits : 4G)					
processing_system7_0	S_AXI_HP0	HP0_DDR_LOWOCM	0x0000_0000	512M	0x1FFF_FFFF
Data_S2MM (32 address bits : 4G)					
processing_system7_0	S_AXI_HP0	HP0_DDR_LOWOCM	0x0000_0000	512M	0x1FFF_FFFF

Fig. 4.13 Address assignment of the design

The block design in Fig. 4.12 was successfully synthesised and implemented in the Vivado IDE. The implementation reports stated that the design can meet the timing constrains and reported the power analysis as shown in Fig. 4.14. Subtracting the “PS7” power consumption from the total dynamic consumption to get the PL (which only has the DPD IP and the essential interfaces) power consumption, the estimated dynamic power consumption for the DPD is 156 mW, which is pretty low.

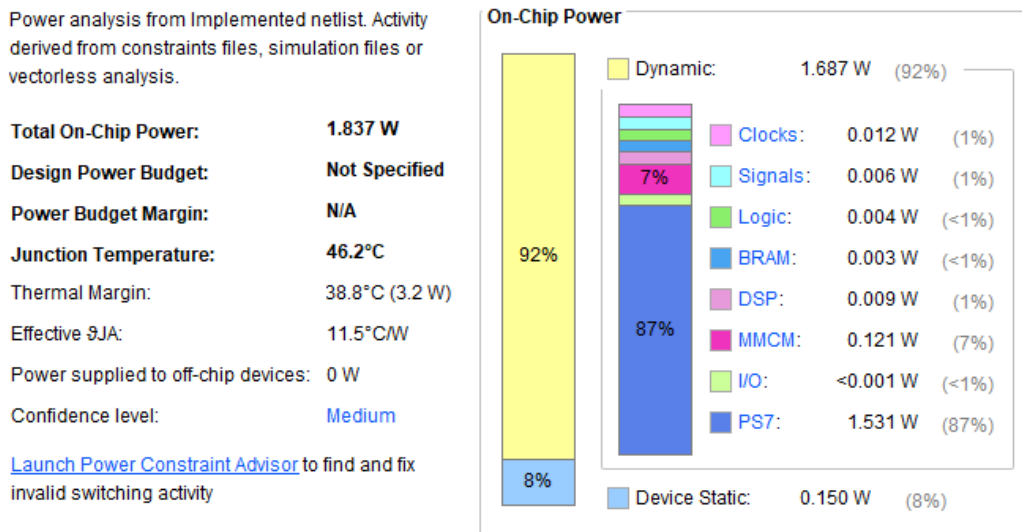


Fig. 4.14 Power analysis of the implementation

CHAPTER 5. CONCLUSIONS AND FUTURE WORK

The main objective of this master thesis has been the optimization of the wireless communication system for UAS in terms of power efficiency. First, we analysed the existing drone communication systems to later proposed an automatic transmit power control mechanism to save unnecessary power consumption. An extended Kalman filter was designed to estimate the required transmit power or the transmitter gain. Given the number of different applications and thus possible payloads in UAS, real-time applications are not limited to sending data from a RGB camera but may include data from other applications such as LiDAR or infrared camera for example, where low latency communications with the GCS is of importance. This project designed a custom OFDM downlink to satisfy the need of sending any kind of data regardless of the data source. By modulating the data of an image and simulating the transmission through a multipath channel model, we have seen and evaluated the effect of linear distortion caused by the communications channel.

In a more hardware-oriented approach (and thus closer to reality), we built a PA testbed based on the Pluto-SDR platform. The SDR platform can up-convert the baseband data to RF signal and down-convert the RF signal to baseband data. The power supply of the PA can be tuned through remotely controllable interfaces, so it is a fully functional testbed for measuring the PA characteristic and evaluating the proposed LUT-based DPD. After conducting an intensive search to find not only the best PA polarization (by tuning the power supply settings) but also the most convenient mean input power (by tuning the transmitter gain), and including DPD linearization, the project finally reached the design goal of having the best power efficiency for different output power levels while meeting the required linearity specifications. Despite the fact that, due to the pre-amplifier gain limitations we could not push the PA to deliver higher output power levels where the power efficiency is significantly higher, the searching strategy presented in this thesis has been proved to be effective.

Finally, the possibility of implementing such system in the AP-SoC FPGA platform was proved. By using the HLS method, the functionality of the IFFT IP core from the HLS library was validated. Then, we proposed a LUT-based DPD architecture, whose coefficients can be derived from memory-based DPD polynomials models such as the GMP. The throughput performance, the resource usage and the estimation of PL power consumption were obtained and discussed. The experimental results showed that the proposed LUT-based DPD can be implemented in the Xilinx Zedboard, which is an FPGA platform suitable to be used on an UAS platform.

For future works, more building blocks of the communication system obviated in this thesis can be further designed. For example, taking more consideration on the payload data modulation, streaming protocol, data compression method, etc. In addition, from the hardware perspective, the integration of the PA, transceiver and FPGA in the UAS platform remains as pending work. A complete wireless communication system is such a large topic that requires persistent development. Wireless communication systems and unmanned aircraft system are naturally

paired, a better design of the communications system can impact positively in several UAS applications.

REFERENCES

- [1] Digikey: Typical ISM Transceiver ICs, 2020, <https://www.digikey.com/products/en/rf-if-and-rfid/rf-transceiver-ics/879>.
- [2] *Alsharqa, A.*: Energy Management in Cellular HetNets Assisted by Solar Powered Drone Small Cells. *In: 2017 IEEE Wireless Communications and Networking Conference (WCNC), Wireless Communications and Networking Conference (WCNC), 2017 IEEE (2017)*, S. 1.
- [3] EASA: Commission Delegated Regulation (EU) 2019/945 of 12 March 2019 on unmanned aircraft systems and on third-country operators of unmanned aircraft systems, 2019, https://eur-lex.europa.eu/eli/reg_del/2019/945/oj.
- [4] *Dan, M.; Cornelis, S.*: Software Define Radio for Analyzing Drone Communication Protocols. *In: , 2018*, S. 485-490.
- [5] DJI: Specification of DJI Phantom, 2020, <https://www.dji.com/uk/products/compare-consumer-drones>.
- [6] *Al-Sa'd, M.F.; Al-Ali, A.; Mohamed, A. et al.*: RF-based drone detection and identification using deep learning approaches – An initiative towards a large open source drone database. *In: Future Generation Computer Systems 100 (2019)*, S. 86-97.
- [7] *Muhammad Usman, S.; Fayezeah, G.; Kalle, R. et al.*: Drone detection and classification using cellular network A machine learning approach. *In: 2019 IEEE 90th Vehicular Technology Conference (VTC2019-Fall) (2019)*.
- [8] DJI: DJI Lightbridge Specs, 2020, <https://www.dji.com/uk/lightbridge-2/info#specs>.
- [9] *Rong, X.L.I.; Member, S.; Vesselin Jilkov, I.P.*: Survey of Maneuvering Target Tracking Ausgabe 2003.
- [10] *Yong Soo, C.*: MIMO-OFDM Wireless Communications with MATLAB®, 2010.
- [11] *Yan, C.; Fu, L.; Zhang, J. et al.*: A Comprehensive Survey on UAV Communication Channel Modeling (2019).
- [12] *Bing, L.*: Study on Modeling of Communication Channel of UAV. *In: 107 (2017)*.
- [13] *López, D.; Gilabert, P.L.; Montoro, G. et al.*: Peak cancellation and digital predistortion of high-order QAM wideband signals for next generation wireless backhaul equipment (2014).
- [14] *Kim, W.-J.; Cho, K.-J.; Stapleton, S.P. et al.*: An efficient crest factor reduction technique for wideband applications. *In: Analog Integrated Circuits and Signal Processing 51 (2007), Heft 1*, S. 19-26.

- [15] *Hsieh, H.; Wu, W.*: Maximum Likelihood Timing and Carrier Frequency Offset Estimation for OFDM Systems With Periodic Preambles. *In: IEEE Transactions on Vehicular Technology* 58 (2009), Heft 8, S. 4224-4237.
- [16] *Daffara, F.; Adami, O.*: A new frequency detector for orthogonal multicarrier transmission techniques. *In: 2* (1995).
- [17] *Moose, P.H.*: A technique for orthogonal frequency division multiplexing frequency offset correction. *In: IEEE Transactions on Communications* 42 (1994), Heft 10, S. 2908-2914.
- [18] *Agarwal, A.; Kumar, B.; Agarwal, K.*: BER Performance Analysis of Image Transmission Using OFDM Technique in Different Channel Conditions Using Various Modulation Techniques. *In: , 2019*, S. 1-8.
- [19] *Cripps, S.C.*: RF power amplifiers for wireless communications. 2nd ed, 685 Canton Street Norwood, 2006.
- [20] *Vassilakis, B.; Cova, A.*: Comparative analysis of GaAs/LDMOS/GaN high power transistors in a digital predistortion amplifier system. *In: 2* (2005).
- [21] *Jarrett, L.*: SWaP – The RF Solution That Can Mean the Difference Between Flying High and Being Grounded, 2015, <https://www.analog.com/en/technical-articles/swap-the-rf-solution.html>.
- [22] *Lei, D.; Zhou, G.T.; Morgan, D.R. et al.*: Memory polynomial predistorter based on the indirect learning architecture. *In: 1* (2002).
- [23] *Morgan, D.R.; Ma, Z.; Kim, J. et al.*: A Generalized Memory Polynomial Model for Digital Predistortion of RF Power Amplifiers. *In: IEEE Transactions on Signal Processing* 54 (2006), Heft 10, S. 3852-3860.
- [24] *Mondal, R.; Ristaniemi, T.; Doula, M.*: Genetic Algorithm Optimized Memory Polynomial digital pre-distorter for RF power amplifiers (2013).
- [25] *Zhu, A.*: Decomposed Vector Rotation-Based Behavioral Modeling for Digital Predistortion of RF Power Amplifiers. *In: IEEE Transactions on Microwave Theory and Techniques* 63 (2015), Heft 2, S. 737-744.
- [26] *Kantana, C.; Venard, O.; Baudoin, G.*: Comparison of GMP and DVR models (2018).
- [27] ADI: ADALM-PLUTO Overview, 2020, <https://wiki.analog.com/university/tools/pluto> [Zugriff am: 13.05.2020].
- [28] *Julieta, F.*: Design and linearization of an efficient class-E power amplifier using a test bench based on development boards, Master, 2017.
- [29] *Gilabert, P.L.; Montoro, G.; Bertran, E. et al.*: FPGA-based set-up for RF power amplifier dynamic supply with real-time digital adaptive predistortion (2010).
- [30] Xilinx, Inc.: Using the MicroBlaze Processor to Accelerate Cost-Sensitive Embedded System Development (WP469).

- [31] ARM: Arm MPS3 FPGA Prototyping Board, 2020, <https://developer.arm.com/tools-and-software/development-boards/fpga-prototyping-boards/mps3>.
- [32] Xilinx, Inc.: Zynq-7000 SoC Technical Reference Manual (UG585) (2018).
- [33] Xilinx, Inc.: UltraFast Embedded Design Methodology Guide (UG1046) (2018).
- [34] *Crockett, L.H.; Elliot, R.A.; Enderwitz, M.A. et al.*: The Zynq Book: Embedded Processing with the Arm Cortex-A9 on the Xilinx Zynq-7000 All Programmable Soc. Strathclyde Academic Media, Glasgow, GBR, 2014.
- [35] Xilinx, Inc.: 7 Series DSP48E1 Slice User Guide (UG479), 2018.
- [36] Xilinx, Inc.: 7 Series FPGAs Memory Resources User Guide, 2019.
- [37] The Zedboard, <http://zedboard.org/product/zedboard>.
- [38] Xilinx, Inc.: Vivado Design Suite User Guide: High-Level Synthesis (UG902), 2018.
- [39] Xilinx, Inc.: Reduce Power and Cost by Converting from Floating Point to Fixed Point (WP491).
- [40] *Akbarpour, B.; Tahar, S.; Dekdouk, A.*: Formalization of Fixed-Point Arithmetic in HOL. *In: Formal Methods in System Design 27* (2005), 1-2, S. 173-200.
- [41] ARM Limited: AMBA AXI and ACE Protocol Specification AXI3, AXI4, and AXI4-Lite ACE and ACE-Lite.
- [42] ARM Limited: AMBA 4 AXI4-Stream Protocol Specification.
- [43] Xilinx, Inc.: Fast Fourier Transform v9.1 LogiCORE IP Product Guide.
- [44] Albert Cesari, Pere L. Gilabert, Eduard Bertran, Gabriel Montoro, Jean M. Dilhac: European Microwave Integrated Circuit Conference, 2007 – EuMIC 2007 ; 8 - 10 Oct. 2007, Munich, Germany ; [part of] European Microwave Week 2007, EuMW 2007. European Microwave Association; European Microwave Integrated Circuit Conference; European Microwave Integrated Circuits Conference *et al.* IEEE, Piscataway, NJ, 2007.
- [45] *Guan, L.; Zhu, A.*: Low-Cost FPGA Implementation of Volterra Series-Based Digital Predistorter for RF Power Amplifiers. *In: IEEE Transactions on Microwave Theory and Techniques 58* (2010), Heft 4, S. 866-872.
- [46] *Younes, M.; Hammi, O.; Kwan, A. et al.*: An Accurate Complexity-Reduced “PLUME” Model for Behavioral Modeling and Digital Predistortion of RF Power Amplifiers. *In: IEEE Transactions on Industrial Electronics 58* (2011), Heft 4, S. 1397-1405.
- [47] N. Safari; P. Fedorenko; J. S. Kenney *et al.*: Spline-Based Model for Digital Predistortion of Wide-Band Signals for High Power Amplifier

Linearization. *In*: : 2007 IEEE/MTT-S International Microwave Symposium, 2007, S. 1441-1444.

[48] Jinja2, <https://jinja.palletsprojects.com/en/2.11.x/>.