Instituto Politécnico de Leiria
Escola Superior de Tecnologia e Gestão
Electrical Engineering Department
Master's Degree in Electrical Engineering
Electronics and Telecommunications Field

MILLIMETER WAVE SOFTWARE DEFINED RADIO

JOÃO GONÇALO CRUZ SILVA

Leiria, March 2025

Instituto Politécnico de Leiria
Escola Superior de Tecnologia e Gestão
Electrical Engineering Department
Master's Degree in Electrical Engineering
Electronics and Telecommunications Field

DISSERTATION

MILLIMETER WAVE SOFTWARE DEFINED RADIO

JOÃO GONÇALO CRUZ SILVA
Number: 2222901
E-Mail: 2222901@my.ipleiria.pt

Dissertation performed under the supervision of Prof. Doutor Luís Miguel Moreira Mendes (lmendes@ipleiria.pt) and Prof. Doutor João Caldinhas Vaz (joaovaz@tecnico.ulisboa.pt).

Leiria, March 2025

Bureaucracy destroys initiative. There is little that bureaucrats hate more than innovation, especially innovation that produces better results than the old routines.

— Frank Herbert

ACKNOWLEDGMENTS

Undertaking this research has been both a challenging and rewarding experience. Throughout this process, I have been fortunate to receive guidance, encouragement, and support from numerous individuals and institutions, without whom this work would not have been possible. I would like to take this opportunity to express my sincere gratitude to all those who have contributed to the completion of this thesis.

I am profoundly grateful to my supervisors, Prof. Luís Mendes and Prof. João Vaz, for their invaluable guidance and encouragement throughout this journey. Their unwavering support and insightful feedback pushed me to constantly strive for more and achieve better results. Their mentorship not only shaped the direction of this work but also helped me grow both academically and professionally.

A heartfelt thank you to my parents and my girlfriend, whose unwavering support, patience, and belief in me made this endeavor possible. Their encouragement and understanding provided the strength I needed during the most challenging moments, and their presence has been a constant source of motivation.

I am also grateful to Prof. Carlos Fernandes and Prof. João Felício from Instituto de Telecomunicações' Antennas and Propagation - LX Group for generously providing access to essential laboratory instruments and mmWave antennas. These resources were crucial for the successful development of the work presented here, given that without those instruments, practical evaluation of the mmWave circuits would not be possible. Valuable resources from IPLeiria's School of Technology and Management have also been utilized during this work, which I would like to show appreciation for.

A special thanks to Instituto de Telecomunicações and the Wireless Circuits - LX Group for granting me this research opportunity and for providing me with the necessary equipment to conduct this study.

ABSTRACT

This study presents the design, implementation, and evaluation of an advanced SDR platform that significantly enhances flexibility, performance, and expandability in wireless communication systems. The research encompasses the development of a base SDR platform operating from 100 MHz to 6 GHz, complemented by a mmWave expansion card extending the frequency range up to 40 GHz. The platform's architecture integrates a powerful signal processing core, featuring both FPGA-based hardware processing and a quad-core ARM CPU, enabling efficient handling of complex signal processing tasks.

Key features of the developed SDR platform include a wide frequency range, high bandwidth capabilities of up to 56 MHz, and flexible sampling rates up to 60 MSPS (with potential for 120 MSPS). The system incorporates MIMO 2x2 functionality and an on-board frequency synthesizer operating up to 18 GHz, enhancing its versatility for various wireless applications. The platform's expandability is demonstrated through the successful integration of the mmWave expansion card, showcasing its readiness for next-generation wireless technologies, including 5G and beyond. Additionally, the SDR's modular design approach, coupled with multiple power options, such as USB-PD and a high-speed PCIe interface for data offloading, ensures adaptability to diverse research scenarios and future technological advancements.

Extensive laboratory and real-environment assessments validate the SDR platform's performance, confirming its compliance with and, in some instances, surpassing initial design specifications. The research highlights the platform's potential for exploring new modulation schemes, investigating spectrum sharing techniques, and developing novel wireless protocols. This work significantly contributes to the field of software defined radio, providing a robust foundation for future research in wireless communications, from IoT to fixed and cellular communications and radar systems.

Keywords: SDR, mmWave, DSP, PCB, wireless.

C O N T E N T S

LIST OF FIGURES

LIST OF TABLES

ACRONYMS

2G	second-generation.
3G	third-generation.
4G	fourth-generation.
5G	fifth-generation.
ACPR	adjacent channel power ratio.
ADC	analog-to-digital converter.
AGC	automatic gain control.
API	application programming interface.
ARPANET	Advanced Research Projects Agency Network.
ASIC	application specific integrated circuit.
AXI	Advanced eXtensible Interface.
BAR	base address register.
BER	bit error rate.
BLE	Bluetooth Low Energy.
BOM	bill of materials.
BPF	band-pass filter.
BPSK	binary phase shift keying.
CAGR	compound annual growth rate.
CDC	clock domain crossing.
CIC	cascaded integrator-comb.
CM4	compute module 4.
CMOS	complementary metal-oxide semiconductor.
CODEC	coder / decoder.

COTS	commercial off-the-shelf.
CPU	central processing unit.
CPWG	co-planar waveguide.
CSI	camera serial interface.
CUDA	compute unified device architecture.
DAC	digital-to-analog converter.
DDR	double data rate.
DDS	digital direct synthesizer.
DMA	direct memory access.
DSI	display serial interface.
DSP	digital signal processing.
EM	electromagnetic.
EMI	electromagnetic interference.
ESR	equivalent series resistance.
ETSI	European Telecommunications Standards Institute.
EVM	error vector magnitude.
FCC	Federal Communication Commission.
FDD	frequency-division duplex.
FEC	forward error correction.
FFT	fast Fourier transform.
FIFO	first-in first-out.
FIR	finite impulse response.
FOM	figure of merit.
FPGA	field-programmable gate array.
FPU	floating-point unit.
FSM	finite state machine.

GPIO general purpose input/output.
GPSDO GPS disciplined oscillator.
GPU graphics processing unit.
GSG ground-signal-ground.
GSM Global System for Mobile Communications.

HDL hardware describing language.
HDMI high-definition multimedia interface.
HLS high-level synthesis.
HSPA High Speed Packet Access.

I2C inter-integrated circuit.
I2S inter-integrated sound.
IC integrated circuit.
IF intermediate frequency.
IFFT inverse fast Fourier transform.
IIR infinite impulse response.
IoT internet of things.
IP intellectual property.
ISA instruction set architecture.

LDO low dropout.
LDPC low-density parity check.
LNA low noise amplifier.
LO local oscillator.
LPF low-pass filter.
LTE Long Term Evolution.
LUT look-up table.

MCU micro-controller unit.

MER	modulation error rate.
MIG	memory interface generator.
MIMO	multiple-input multiple-output.
mmWave	millimeter-wave.
MPEG	Moving Picture Experts Group.
MSI	message signaled interrupt.
NCO	numerically controlled oscillator.
OFDM	orthogonal frequency division multiplexing.
O-RAN	Open Radio Access Network.
PA	power amplifier.
PC	personal computer.
PCB	printed circuit board.
PCIe	peripheral component interconnect express.
PDN	power delivery network.
PFD	phase-frequency detector.
PI	power integrity.
PLL	phase-locked loop.
PMBus	power management bus.
PMC	power management controller.
PMU	power management unit.
PN	pseudo-noise.
PSRR	power supply rejection ratio.
QAM	quadrature amplitude modulation.
QPSK	quadrature phase shift keying.
R&D	research and development.

RAM	random access memory.
RF	radio frequency.
RISC	reduced instruction-set computing.
RRC	root-raised cosine.
SBC	single board computer.
SDR	software defined radio.
SFDR	spurious-free dynamic range.
SIMD	single instruction-multiple data.
SoC	system-on-chip.
SoM	system-on-module.
SPI	serial peripheral interface.
SSB	single sideband.
SWaP-C	size, weight, power consumption and cost.
TCXO	temperature-compensated crystal oscillator.
TDD	time-division duplex.
TDM	time-division multiplexing.
TLP	transaction layer packet.
TS	transport stream.
UMTS	Universal Mobile Telecommunications Service.
USB	universal serial bus.
USB PD	universal serial bus power delivery.
V2I	vehicle-to-infrastructure.
V2V	vehicle-to-vehicle.
VCO	voltage controlled oscillator.
VNA	vector network analyzer.

WCDMA	Wideband Code-division Multiple Access.
XIP	execute in-place.

1

INTRODUCTION

The rapid evolution of wireless communication technologies has driven an unprecedented demand for high-bandwidth, low-latency, and adaptable systems. As the field progresses into the era of fifth-generation (5G) and beyond, millimeter-wave (mmWave) frequencies have emerged as a promising solution to meet these requirements. Concurrently, software defined radio technology has transformed the landscape of radio communications by providing exceptional flexibility and adaptability.

The convergence of wireless communication advancements and the evolution of radio and electronics has motivated this research and development work. In the following sections, a concise history of radio communications is presented to contextualize this work. The motivation leading to the definition of the research objectives is described, along with the objectives themselves. Additionally, the document structure and highlights of the scientific contributions made through this study are presented.

1.1 RADIO COMMUNICATION BRIEF HISTORY

Since 1890, when the first wireless transmissions took place, radios have experienced a continuous evolution, providing users the possibility to stay connected with increasing transmission rates. Around mid 1930, band limited analog voice communications saw great use (e.g., telephone lines). This was the first big milestone in radio communication systems. The second world war also contributed extensively to the development of communication systems, presenting updated technology that allowed messages to be delivered and sent faster than ever before, and with higher levels of security. In the 50s, analog television broadcasting occupied even more bandwidth, in order to provide a better customer experience.

As miniaturization of computers began to happen, in the 60s, they started to show interesting use cases for communication across long distances. Wired connectivity was established via the United States Advanced Research Projects Agency Network

(ARPANET), which later became the Internet. Wireless connections were facilitated by satellites, as in 1962, Bell Labs launched the first active orbiting communications satellite, offering yet another field of use for radio communications [1].

At the end of the 70s, cellular networks started to emerge, allowing people to wirelessly communicate through voice from any public vehicle. The first cell phones, big and bulky devices, provided to be hard to operate, not like modern smartphones, which are effectively tiny pocket computers. These allow anyone who carries them to connect with the rest of the world at speeds that were unimaginable a generation ago.

In 1985, the United States Federal Communication Commission (FCC) officially established the concept of unlicensed spectrum for industrial (902-928 MHz), scientifical (2400-2483.5 MHz) and medical (5725-5850 MHz) (ISM) purposes, which led to the creation of short range wireless communication standards, such as WiFi and Bluetooth, which are now a part of daily communications media.

Even though multiple technologies have proven to be effective and achieved large growth, there is one issue common to all of them: their radios and protocols are mostly implemented through hardware based solutions, which offer little to no programmability nor reconfigurability out of what was originally standardized, and pose a potential problem shall an error occur in hardware, firmware or software. In that case, there is usually no sensible way of correcting the problem. There is also the potential for vulnerabilities, that are discovered many years after standardization of the protocol, to be exploited, since their removal is not easily achievable. In fact, a hardware based radio solution is usually limited to its original functionality, and is not able to operate according to any other wireless protocols beyond that. Software defined radios (SDRs) are one of the concepts that provide a solution to this problem.

The term software defined radio was defined by Dr. Joseph Mitola in the 1990s to refer a radio that can be reconfigured and reprogrammed easily via software, although the concept had been introduced earlier by E-Systems [2]. The importance of SDRs in modern societies cannot be overstated. Mitola's 1992 IEEE publication popularized the term among radio engineers [3]. Since its inception, SDR technology has undergone significant evolution. As of 2025, SDRs play a crucial role in modern societies by enabling flexible and cost-effective wireless communication systems, supporting multiple radio protocols, and facilitating technological advancements.

Like most technologies, SDRs had been used and experimented with in military environments, before being available for the general public [1]. The first operational

SDR, known as Speakeasy [4] was developed by the United States' Navy between 1991 and 1995. Besides the fact that the device occupied the entire backside of a military transport vehicle, it could not be used for any other purpose than what the hardware was originally built for. Its successor, Speakeasy II, was more successful and reusable, mainly due to advances in electronics and wireless communication circuits.

Until around the mid-1990s, virtually all commercial radio frequency (RF) transceivers were analog intensive, designed from the ground up with a specific task in mind, and offering little to no configuration options at all. In the past two decades, however, such analog-intensive radio transceivers have been evolving to allow their reconfigurability, becoming more digitally intensive. RF performance of highly integrated digital transceivers has been increasing continuously ever since, due to the changing nature of RF circuit design. RF transceivers are going through the same process which affected the field of analog audio in the 1980s and 1990s, when innovative digital processing techniques (e.g., oversampling, noise shaping, calibration, etc...) started being implemented [5].

Nowadays, the use of SDRs in the microwave and millimeter-wave bands have become increasingly prominent and impactful. These advanced communication systems exploit the high-frequency spectrum, enabling the transmission of large amounts of data at remarkable speeds. With their broad bandwidth and enhanced capacity, mmWave SDRs enable faster and more reliable wireless connections, facilitating the realization of emerging technologies like 5G and beyond. Moreover, the adaptability and reconfigurability of SDRs make them an ideal platform for prototyping and testing new mmWave applications, fostering advancements in wireless technology such as in Open Radio Access Networks (O-RANs). References [6–8] present some examples of the usage of SDRs at mmWave frequencies, for a variety of purposes. As the demand for high-speed, low-latency, and high-capacity wireless communication continues to grow, mmWave SDRs hold immense promise for revolutionizing connectivity across various industries and driving the next generation of wireless innovation.

1.2 MOTIVATION

Over the past decade, high-frequency radios with wideband capabilities, high dynamic range, spectral agility, modulation-agnostic features, and support for multiple standards have garnered significant attention from multiple fields for their

diverse range of applications. Simultaneously, recent technological advancements have allowed architecture changes to radios that shrink their size, weight, power consumption and cost (SWaP-C), while maintaining performance and evolving toward software-programmable common hardware, that is, evolving to SDRs, which are now considered as the future of mobile communications [9]. The operational flexibility of this type of radio allows them to be used in various fields of activity, such as telecommunications, transportation, aerospace, and defense, among other potential applications.

A software defined radio is a type of radio system where components that have been traditionally implemented by fixed-function electronic circuits, such as mixers, filters, modulators and demodulators, are, instead, defined through software. As though it may seem, a digital radio is not necessarily a software defined radio, nor is one that can switch between modulation schemes through digital control. A modern cellular phone may support multiple standards, such as Global System for Mobile Communications (GSM) [second-generation (2G)], Wideband Code-division Multiple Access (WCDMA) [third-generation (3G)], Long Term Evolution (LTE) [fourth-generation (4G)] and even 5G. It is, internally, through automatic software control, responsible for selecting the most appropriate standard to use at a given moment in time. This defines the phone as a software-controlled radio, and not a software defined radio [10]. For a radio device to be considered software-defined, it must specifically implement a significant part of the signal processing tasks in the digital domain, either software or firmware [11]. The FCC defines software defined radio as a "generation of radio equipment that can be programmed quickly to transmit and receive on any frequency within a wide range of frequencies, using virtually any transmission format and any set of standards".

One of the most significant advantages of SDRs is their ability to evolve with technology and architecture. With software defined radios, users can update the radio's firmware with newer and more advanced signal processing algorithms, making it possible to adapt to different communication systems and standards. As technology continues to advance, SDRs will continue to progress with faster processors, enhanced security, and more sophisticated algorithms.

The applications of SDRs are vast and range from improved wireless communication technologies to critical industries and fields. For example, in the defense sector, SDRs are crucial in providing reliable and secure communication channels for military forces, as well as in the development of multiband radar that requires wideband, high dynamic range, and agile spectral monitoring devices. Electronic

warfare also benefits from SDRs due to their ability to rapidly adapt to changing signal environments and identify security threats.

With the ever increasing number of devices connected to the Internet, to form an internet of things network, the need to adjust the network arises, in order to accommodate the large collection of endpoints. A large number of wireless protocols have been developed, such as ZigBee, Bluetooth Low Energy (BLE), Thread, and multiple WiFi standards, each one offering different characteristics (e.g., data rate, latency, complexity), thus, each one having its specific use cases [12]. The flexibility of having a single internet of things (IoT) device capable of communicating using that vast collection of protocols, switching between them on-the-fly, as its environment changes, is a commodity made possible by SDRs.

In the transportation sector, SDRs can be integrated into vehicles to enable better communication and navigation systems and, in the automotive industry, SDRs have the potential to revolutionize vehicle-to-vehicle (V2V) and vehicle-to-infrastructure (V2I) communication systems, improving safety, traffic flow, and passenger comfort.

SDRs are particularly relevant in the development of telecommunication equipment and systems, which require immense flexibility to adapt to new standards and protocols in order to meet the changing demands of users. With the rapidly evolving development of wireless protocols and the increasing popularity of multi-protocol compatible devices, the SDR paradigm has proven to be very promising, especially in 5G cellular networks and beyond (e.g., 6G). The SDR enables the development and implementation of new networks and frequency usage without requiring changes to hardware and allows for rapid deployment of new protocols and enhancements that enable much faster data download and upload speeds with low latency.

As the number of mobile communication standards increases rapidly, offering new, faster, and improved services to the users, typical multimode radios that use multiple RF frontend blocks are increasing in cost and complexity, driving up the cost of the user terminal equipment. The use of such fixed architecture of RF frontends makes the radio less flexible and unable to accommodate future revisions of communication standards, or even new standards at all. An SDR would easily be able to receive updates as standards evolve, as it is currently done to other components of user terminal equipment. This concept can be equally applied to the mobile network operator infrastructure (e.g., cellular base stations), as migrations (e.g from Universal Mobile Telecommunications Service (UMTS) to High Speed Packet Access) could be easily accomplished by uploading new software and reconfiguring

the device, without any hardware changes, even though different modulations and frequencies may be used.

SDRs are also essential in industry and manufacturing, where they can serve as a reliable and cost-effective solution for adapting communication systems to changing standards and protocols without the need for changing any hardware or production machinery. Edge computing, and consequently Industry 4.0, can benefit from the usage of SDRs since they perform signal processing in software, making them well-suited for use in edge computing environments where resources are limited and latency is a concern. Additionally, SDRs can be used to perform a wide range of signal processing tasks, including modulation recognition, channel estimation, and spectral analysis, making them useful tools for edge computing applications like wireless network optimization and industrial automation.

Even in the research and development (R&D) field, regardless of whether it is in an academic or industry environment, SDRs are becoming increasingly important due to their flexibility and agility in adapting to new communication standards and protocols. In fields such as electrical and electronic engineering, telecommunications, and computer science, SDRs are enabling researchers to design, emulate, and test a wide variety of wireless communication systems. By using SDRs, researchers can rapidly prototype and evaluate communication systems in real-time, which can be especially helpful in the design phase of a project. In addition to their versatility, SDRs are also important for scientific research due to the potential for cost savings, both in terms of hardware and software. Because SDRs allow for the development of common hardware that can be programmed to fit a wide range of communication standards, researchers can potentially avoid the need for costly specialized hardware. Additionally, by taking advantage of open-source SDR software support libraries, researchers can collaborate and build upon each other's work, further reducing costs and speeding up research progress.

The development and implementation of SDRs in scientific research is important because it enables researchers to stay at the forefront of rapidly evolving communication technologies. As new communication standards emerge and existing ones are updated, SDRs provide a platform for studying and developing new systems. Furthermore, by using SDRs, researchers can gain insights into how communication systems can be optimized for different applications. Overall, the continued study, development, and implementation of SDRs in the context of scientific research is vital for staying ahead of the curve in an ever-changing field of communication technology.

Although a software defined radio can appear to bring only advantages, when compared to traditional fixed radios, there are some disadvantages associated with the concept, such as lower dynamic range and higher power consumption. Besides that, and as usually happens before any technology is standardized, programming an SDR and maintaining the different pieces of software required can be a non-trivial task. In recent years, however, several software solutions have appeared, aiming to standardize both digital signal processing (DSP) [13–15] and the interface [16] with the software defined radio hardware.

As SDRs continue to evolve, they are expected to make use of higher and higher frequencies, including the mmWave band. While there are currently several SDRs available in the market that operate at the lower end of the gigahertz frequency spectrum, the trend is moving towards higher frequencies in order to accommodate new and emerging communication technologies such as 5G and complex radar systems. The mmWave bands offer a much larger bandwidth than traditional radio bands, allowing for much faster data download and upload speeds with low latency. Additionally, these bands have a large amount of unused spectrum, including unlicensed bands, which can be used for new wireless protocols and applications.

Millimeter-wave bands are well-suited for short-range communications such as indoor wireless networks, IoT devices, and vehicle-to-vehicle communications. Although they have a shorter range than traditional radio bands, this can be mitigated using beamforming and advanced signal processing techniques that are well-suited to SDRs. Another advantage of using mmWave SDRs is the high level of security that can be provided. The high frequency and directional nature of mmWave signals make them harder to intercept and jam, which can improve overall security for wireless communication systems.

Overall, the evolution of SDRs towards millimeter-wave bands has the potential to greatly enhance wireless communication capabilities and enable new applications in a wide range of fields, such as autonomous vehicles, smart infrastructure, and the internet of things. These fields require low latency, high bandwidth, and reliability.

According to several market research reports, the global software defined radio market will grow in the coming years. The market is being driven by factors such as the increasing adoption of SDRs in military and defense applications, the rising demand for advanced communication systems (e.g., 5G and future 6G), and the growing need for spectrum-efficient and cost-effective radio solutions. According to a report by MarketsandMarkets, the SDR market is expected to grow from 10 billion USD in 2022 to 12.5 billion USD by 2027, at a compound annual growth

rate (CAGR) of 4.6% during the forecast period [17]. Another report by Fortune Business Insights predicts the SDR market will grow from 11.60 billion USD in 2021 to 16.20 billion USD in 2028 at a CAGR of 5.3% during forecast period [18]. Overall, there appears to be a very positive outlook for the growth of the SDR market in the coming years [19].

1.3 OBJECTIVES

The objective of this work is to explore the architectures and potential implementations of software defined radios, and to develop, implement, and test an SDR capable of operating up to, and including, the millimeter-wave frequency region.

This thesis explores the convergence of the mmWave and SDR cutting-edge technologies through the development of a mmWave SDR. By combining the high-bandwidth capabilities of mmWave frequencies with the versatility of SDR platforms, this research aims to contribute to the advancement of next-generation wireless communication systems.

Through this research, this thesis aims to contribute to the growing field of advanced communication systems and provide valuable insights into the capabilities of software defined radios in mmWave applications and their design and implementation.

1.4 DOCUMENT STRUCTURE

This thesis comprehensively addresses the development of a mmWave capable SDR, encompassing the design goals, circuit schematic design, printed circuit board (PCB) layout, firmware and software of the entire system.

This document is organized in seven chapters. Chapter 2 provides a comprehensive literature review of digitally intensive RF transceivers and presents the SDR system architecture developed in light of this review. The chapter begins by exploring fundamental concepts of digitally intensive RF transceivers, including frequency conversion, signal domain conversion, digital signal processing, and key performance parameters. It then examines various receiver topologies such as direct conversion (homodyne), heterodyne, and direct RF sampling. The chapter also discusses transmitter topologies, including two-step conversion, direct launch, and RF digital-to-analog converters (RF digital-to-analog converters (DACs)). Digital

processing options are compared, covering general-purpose central processing units (CPUs), graphics processing units (GPUs), field-programmable gate arrays (FPGAs), DSPs, and application specific integrated circuits (ASICs). Subsequently, the chapter presents a state-of-the-art review, providing a foundation for understanding design choices and trade-offs in modern RF transceiver architectures, particularly those employing digitally intensive approaches. The chapter concludes with the SDR system architecture and the rationale behind its design choices.

Chapter 3 details the circuit development of the SDR base platform. It begins by outlining the platform architecture, providing an overview of the system's structure. The chapter then delves into the functional block design, covering key components such as the signal processing core, RF transceiver, mmWave synthesizer, clock management, and power management systems. Special attention is given to the expansion card interconnections, highlighting the platform's modularity. The chapter concludes with a discussion of the PCB stackup and layout, addressing the physical implementation of the designed circuits. This comprehensive exploration of the SDR base platform's hardware development forms the foundation for the system's capabilities and performance, setting the stage for subsequent chapters on firmware, software, and the mmWave expansion card.

Chapter 4 focuses on the firmware and software development for the SDR base platform, detailing the integration of hardware and software components. It begins with the FPGA design architecture, outlining the structure and operation of the programmable logic. The chapter then introduces custom intellectual property (IP) cores, such as the Advanced eXtensible Interface (AXI) inter-integrated sound (I2S) Engine, AXI serial peripheral interface (SPI) Engine, AXI RF Timestamping, AXI IRQ Controller, and AXI general purpose input/output (GPIO), which were developed to support specific functionalities. Proprietary IP cores are also discussed, including the PicoRV32 processor, direct memory access (DMA) controller, RF transceiver interface, inter-integrated circuit (I2C) interface, peripheral component interconnect express (PCIe) to AXI bridge, and double data rate (DDR) 3 memory interface. The software section covers key aspects such as firmware for power management control, kernel device driver for addressing hardware components, the userspace SoapySDR driver for interfacing with the SDR platform, and user applications that enable practical usage of the system. This chapter provides a comprehensive overview of the firmware and software components that enable seamless operation of the SDR base platform.

Chapter 5 focuses on the design and development of the mmWave expansion card, a critical component enabling the SDR platform to operate in the mmWave

frequency range. The chapter begins by addressing system design considerations, including the functional architecture of the mmWave card, simulation requirements, and PCB stackup. It then delves into circuit design, covering key aspects such as frequency conversion, internal and external local oscillator (LO) generation and conditioning, LO switching and monitoring, as well as control and monitoring mechanisms. The estimated performance of the mmWave board is discussed next, with attention to PCB layout and performance assessment. Finally, the chapter concludes with an overview of the firmware and software developed to support the mmWave expansion card, ensuring seamless integration with the SDR platform. This chapter provides a detailed exploration of the hardware and software aspects of the mmWave expansion card, emphasizing its role in extending the platform's capabilities.

Chapter 6 presents a comprehensive performance evaluation of the developed SDR system. The chapter begins by assessing the SDR base platform, examining its core functionalities and performance metrics. It then focuses on the mmWave expansion card, providing detailed analysis of its capabilities and performance in the millimeter-wave frequency range. The chapter concludes with a series of demonstration applications, showcasing practical use cases and real-world performance of the complete SDR system. These demonstrations serve to validate the system's functionality and highlight its potential in various applications, providing concrete evidence of the platform's capabilities and the successful integration of the base platform with the mmWave expansion card.

Chapter 7 concludes the thesis by summarizing the key findings and achievements of the research. It reflects on the development of the SDR platform, including both the base system and the mmWave expansion card, and evaluates how well the initial objectives were met. The chapter discusses the significance of the work in the context of current and future wireless communication technologies. Additionally, it outlines potential areas for future R&D, suggesting improvements or extensions to the current system. This final chapter provides closure to the thesis while also defining possible paths for continued exploration in the field of software defined radio and mmWave communications.

1.5 SCIENTIFIC CONTRIBUTIONS

The research and development conducted in this thesis has contributed to the scientific community through the publication of a peer-reviewed paper and open-

source software developments. The publication demonstrate the originality and relevance of the work, addressing key challenges in the development of software defined radios operating at mmWave frequencies.

As far as the author is aware, there currently does not exist a SDR platform that comprises the flexibility and expandability offered by the platform presented in this thesis, while maintaining an affordable cost and reduced size. Other solutions working up to 40 GHz do exist, but at a cost exceeding 150,000 USD.

A conference paper, entitled *Highly Flexible and Scalable Millimeter Wave Software Defined Radio*, was submitted, accepted, and presented at the IEEE 22^{nd} Mediterranean Electrotechnical Conference (MELECON) in 2024, achieving further publication in the conference proceedings. The paper briefly addresses the design procedure of the SDR, presenting practical results obtained on the bench. The paper is available at the end of this thesis, at appendix A. Additionally, this work has also been pitched to the audience of the 17^{th} Congress of the Portuguese URSI Committee, in 2023. A journal paper is also in progress, which will contain the most recent results and findings obtained with the SDR on the bench.

During the development of this SDR, several contributions were made to the open-source digital signal processing library Liquid DSP, by Joseph D. Gaeddert [15], which highly enhanced the performance of this library, while reducing the computational power requirements, by leveraging vector processing instructions present on modern x86/a64 CPUs.

2

LITERATURE REVIEW AND SYSTEM ARCHITECTURE

In the field of modern communications, electronic radios have become a fundamental technology, significantly transforming the transmission and reception of information over vast distances. Over the last few decades, commercial RF transceivers have evolved from analog-intensive designs to more digitally intensive and reconfigurable systems. Today, SDRs show great potential for use in microwave and millimeter-wave bands, enabling high-speed data transmission and supporting emerging technologies like 5G and beyond. Their adaptability makes them ideal for prototyping and testing new millimeter-wave applications, contributing to advancements in wireless technology and promising to revolutionize connectivity across various industries.

This chapter explores the technologies and radio architectures used in modern radios, including SDRs. It examines the extensive scientific and technical literature related to electronic radios, tracing their development from early analog systems to contemporary digital platforms. The chapter investigates the essential principles underlying various radio topologies, with a particular focus on SDR technology. It concludes by outlining the system architecture of the SDR to be developed, taking into account the techniques and topologies discussed previously.

2.1 DIGITALLY INTENSIVE RF TRANSCEIVERS

Digitally intensive RF transceivers represent a significant evolution in wireless communication technology, marking a shift from traditional analog-intensive designs towards more digital implementations. These advanced transceivers leverage the latest developments in digital circuit design and processing capabilities to enhance performance, flexibility, and integration. Key characteristics include increased digital functionality, improved reconfigurability to support multiple communication standards and frequency bands, and better scalability with advancing semiconductor processes [20]. Notably, these transceivers demonstrate reduced sensitivity to process, voltage, and temperature variations compared to their analog counterparts.

The architecture of digitally intensive RF transceivers integrates digital baseband processing, employing sophisticated digital signal processing blocks, and at least one RF transceiver. Key components such as phase-locked loops (PLLs), transmitters, and receivers are implemented with more digital circuitry, enabling improved RF reconfigurability and power efficiency through effective digital power-saving techniques. As depicted in fig. 2.1, a digital intensive RF transceiver fundamentally comprises three main sections, concerning the signal: the frequency conversion stage (also known as the RF frontend), the signal domain conversion and the DSP blocks.

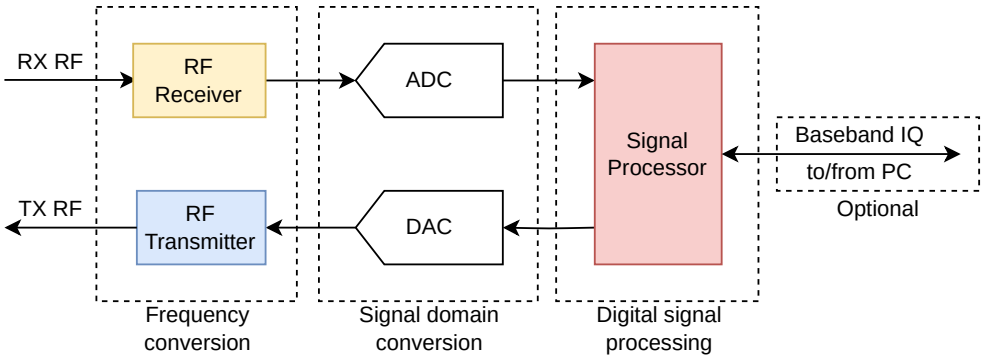


Figure 2.1: Architecture of a digitally intensive RF transceiver.

2.1.1 *Frequency conversion*

The frequency conversion block is the main RF/analog section of the transceiver. In the transmit path, the signal center frequency is shifted from baseband or intermediate frequency into the final desired RF carrier frequency, by a mixer (or up-converter). It may also comprise one or more amplifiers, usually with variable gain control, in order to adjust the output signal power, and filters.

At the receiver side, a low noise amplifier (LNA) is usually present as the first active component in the chain, in order to optimize the overall noise figure of the receiver, as will be explained later. There can also exist a tunable filter present after the LNA, in order to band-limit the received signal. The filtered signal is then shifted into an intermediate frequency (IF) or baseband through a mixer.

The configurability of the frequency conversion stage of a software defined radio is of great importance to the overall flexibility of the transceiver itself. It might not be reprogrammable, in the sense that it is not possible to modify the core behavior of the components (i.e., a mixer will always mix two signals, and it is not possible to replace it with something else), but it should offer the most possible tunable parameters

to the software, in order not to be a bottleneck, such as: high LO frequency range, high power dynamic range and highly configurable filter bandwidths.

2.1.2 *Signal domain conversion*

This section is where the transceiver actually becomes digital. The task of the signal domain conversion stage is to make the transition between the analog and the digital domains. In the transmitter, discrete samples from the digital domain are converted into an analog voltage or current through a DAC, creating an analog representation of the signal. There might also exist a filter after the DAC, in order to suppress unwanted harmonic and spurious contents of the signal before the frequency conversion stage.

In the receiver, the reverse operation is performed. The downconverted signal from the frequency conversion stage is first conditioned, and then sampled and discretized by an analog-to-digital converter (ADC).

Given that this section is also commonly not programmable, it is usually also required that it offers some configurability, such as a wide range of supported sampling frequencies.

2.1.3 *Digital Signal Processing*

Having covered the signal domain transitions, it is now possible to implement almost any kind of signal processing techniques in the digital domain. This section can comprise one or more digital signal processing devices, that should be completely reprogrammable in the field. Commonly implemented digital processing blocks include digital up- and downconverters, whose task is similar to the analog frequency conversion block. They typically comprise one or more numerically controlled oscillators (NCOs), which are used to generate local oscillator tones for the numerical mixers, which work just like analog mixers, and perform the multiplication (complex or real) in order to translate a signal up or down in frequency. There are also digital filters, of which there exist two main types: finite impulse response (FIR) and infinite impulse response (IIR). The cascaded integrator-comb (CIC) filter is a special case of a FIR filter, and stands out in the DSP domain by its simplicity, low resource usage and computational efficiency [21]. There also exist FIR filters capable of implementing mathematical operations, such as derivatives or the Hilbert

transform. The fast Fourier transform (FFT) and it's inverse, the inverse fast Fourier transform (IFFT), are also among commonly implemented algorithms in the digital processing section, used, for example, in orthogonal frequency division multiplexing (OFDM) modulation. It is common for digital transmission protocols to employ some form of forward error correction (FEC) scheme, sometimes even concatenating multiple algorithms in order to offer resilience to the transmitted data. Examples include convolutional codes, turbo codes, low-density parity check (LDPC) codes and Reed-Solomon codes. The decoding process for these algorithms is typically very processing intensive, and is also performed by the digital processing block.

2.1.4 *Key performance parameters*

In order to later compare different existent software defined radio solutions, some key performance indicators are now presented. The RF frontend performance plays a critical role in any radio and it is also the main area of testing when qualifying radio devices for regulatory standards for RF emissions (e.g., FCC, European Telecommunications Standards Institute (ETSI)).

The important characteristics for the analog transmitting frontend are the following:

- *1 dB compression point:* This is the amount of output power the transmitter can produce (OP_{1dB}) - and the corresponding input power (IP_{1dB}) - when its gain is 1 dB below the small-signal gain. The P_{1dB} is a large-signal performance figure of merit (FOM). Ideally, no amplifier should be operated at it's 1 dB compression point, however, there are some modulation techniques that can tolerate the impairments introduced by compressed amplifier operation. The P_{1dB} should be as high as possible, without compromising on power efficiency or space occupied by the device.
- *Third order intercept point:* This is a theoretical small-signal performance parameter that represents the output (OIP3) or input (IIP3) point at which the power of the third order intermodulation products would equal the power of the wanted signal. This power level is not achievable in practice, as the amplifier will reach saturation much before this point. It is obtained by linearly extrapolating (beyond saturation) the curves that correlate the output power of both the fundamental tone and intermodulation products, as depicted in fig. 2.2. As P_{1dB}, the difference between this parameter and the operating power point of an amplifier is a measure of how much distortion will be

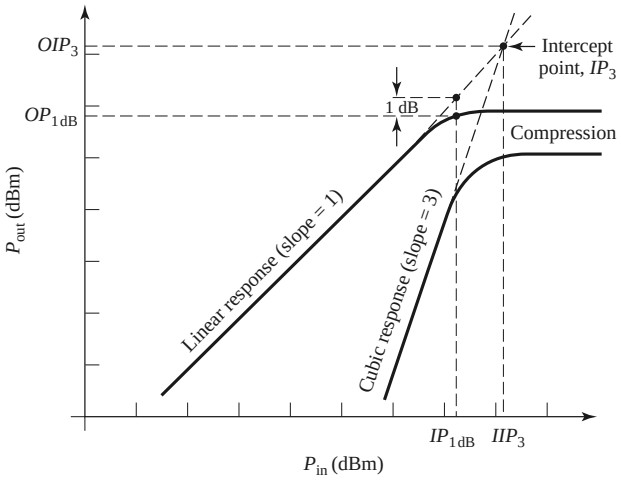


Figure 2.2: Output vs input power of an amplifier, highlighting the P_{1dB} and the theoretical IP3 point [22].

introduced in the signal. The higher the OIP3 of a transmitter frontend, the better. The overall IP3 of two cascaded devices can be given by

$$IP3_C = \left(\frac{1}{IP3_1 G_2} + \frac{1}{IP3_2}\right)^{-1} [mW], \tag{2.1}$$

where $IP3_C$ is the cumulative IP3 (OIP3 or IIP3) of both devices; $IP3_1$ and $IP3_2$ are the individual IP3 value of stage 1 and 2, respectively, and G_2 represents the individual gain of stage 2. This formula can be applied recursively to a chain of N devices to find out its cumulative IP3.

- *Output power dynamic range:* This is a parameter of the transmitting frontend as a whole, and represents the range of transmitted signal power levels. Ideally this should be as high as possible, in order to grant a wide range of possible application fields for the SDR.
- *Spurious emission:* Unwanted signals can be present on the output of an SDR, such as harmonics of the signal, local oscillator leakage tones, or intermodulation distortion that leads to spectral regrowth and constellation deformation, increasing the error vector magnitude (EVM). It is desired that any unwanted component kept as attenuated as possible, in order for the signal to maintain its integrity and avoid interferences with other wireless signals/systems. This is sometimes hard to accomplish, especially when the frontend covers a wide range of frequencies.
- *Frequency stability:* Both long and short-term frequency stability are important. A long-term frequency deviation on the local oscillator can cause the transmitted signal to drift off the receiving end filter bandwidth, or drift onto

a neighboring channel, causing interference. Short term stability, represented by phase noise, can cause in-phase and quadrature (IQ) constellation points to move away from their correct location, worsening the EVM, the modulation error rate (MER), and, consequently, the bit error rate (BER).

As far as the analog receiver frontend is concerned, besides spurious contents and frequency stability, as described for the transmitter, the following are also critical points:

- *Sensitivity:* This is the minimum signal power that has to be present at a receiver's input, while it still being able to discern signal from noise, and it can vary between modulation techniques, since different modulations require different levels of minimum signal-to-noise ratios.
- *Noise figure:* The measure of how the receiver degrades the signal-to-noise ratio present at its input. It is desired that the receiver has the lowest possible noise figure, and this is accomplished, for example, by using a very good LNA, with a low noise figure and high gain, as the first element of the receiving chain, as proven by the Friis formula for noise:

$$F_{RX} = F_1 + \frac{F_2 - 1}{G_1} + ... + \frac{F_n - 1}{G_1 G_2 \ ... \ G_{n-1}}, \tag{2.2}$$

  where F_{RX} is the overall receiver noise figure, F_1 is the noise figure of the first block, usually an LNA, G_1 is the gain of the first block, F_n is the noise figure of the n^{th} block in the receiver, and G_{n-1} is the gain of the $(n-1)^{th}$ block in the receiver.
- *Dynamic range:* This parameter of the receiving frontend represents the range of power levels it can tolerate on its input, while still working correctly. The low end of this range is defined by its sensitivity, whilst, at the high end, it is limited by the overloading of components in the chain or strong signal handling performance. Ideally, a receiver should have the highest possible dynamic range, meaning it can decode faint signals, while also tolerate higher power ones.

As a whole system, several other parameters are considered important for a software defined radio, such as:

- *Sampling rate:* The sampling rate of the data domain conversion stage limits the bandwidth that the radio is able to process, and, consequently, shortens the list of protocols it supports. As per the Nyquist theorem, the sampling frequency should be at least twice the bandwidth of the signal of interest.

- *Flexibility and reprogrammability:* These parameters characterize the radio's capability for the modulation and air-interface algorithms and protocols to be changed by solely loading new software onto the platform. The higher the flexibility and reprogrammability, the more applications an SDR has.
- *Adaptability:* The SDR platform can adjust its capabilities as network or traffic operational requirements change. This means that it has a broad range of application scenarios. For example, a SDR can be used by a mobile phone to connect to terrestrial cellular networks, and the same radio can, moments later, be used to connect to satellites using a totally different modulation scheme and protocol, when terrestrial coverage is lost.
- *Computational power:* The processing rate of the SDR platform, namely operations per second (OPS). More computation power means more intensive algorithms are supported by the platform, such as FEC algorithms.
- *Energy efficiency:* The relation between total power consumption and offered performance. It is desired that an SDR has the best energy efficiency possible, especially when the platform is expected to be operated on battery power.

2.2 RECEIVER TOPOLOGIES

The two most important receiver architectures, described below, are the super-heterodyne receiver, useful for most high-tier broadband and narrowband applications, and the direct conversion receiver, for medium tier broadband applications. Also addressed in this chapter is the more recent direct RF sampling architecture.

2.2.1 *Direct conversion or Monodyne*

The direct conversion receiver, depicted in fig. 2.3, is the form of software defined radio whose architecture is the closest to a typical non-reconfigurable receiver, and is suitable for delivering broadband data with medium performance specifications. In this implementation, there is only one mixing stage involved, which directly converts the signal from RF to baseband. In this type of receiver, the transition from the analog to the digital domain only occurs when the signal is already mixed down to baseband, thus, typically requiring lower processing power available on the digital signal processor. Since the baseband consists of a complex IQ signal, two

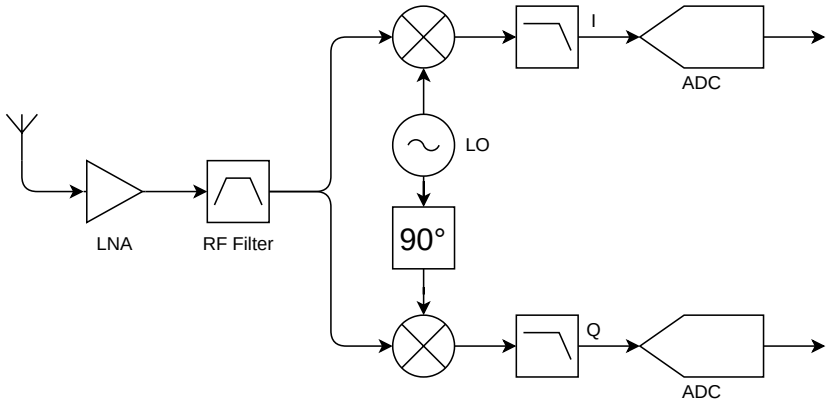


Figure 2.3: Direct conversion receiver architecture.

converters are necessary. In absence of an IF filter, all the protection from close-in interferers must come from the I and Q low-pass filters (LPFs).

With a complex IQ baseband, the analog section does not need to know a priori what type of modulation is being used, neither what kind of processing happens after digital sampling.

Some of the advantages and disadvantages of this architecture are presented below [23].

- Advantages
  - Simpler architecture: only a single LO is required, the frontend can be simplified, and the IF chain is removed;
  - Lower cost;
  - Lower component count and occupied area;
  - Several hardware-driven tasks become software-driven, which makes it easier to dynamically adjust the channel bandwidth, and allows for flexible multimode operation;
  - Signal domain converters run a lower sampling rates, requiring less digital processing power.
- Disadvantages
  - Linearity, spectral purity and IQ balance constraints are tighter than in the superheterodyne topology;
  - Two analog-to-digital converters are needed, one for the I and another for the Q components;

– The lower baseband frequencies are susceptible to various interference sources, such as "1/f" (flicker) noise, DC offset, self-mixing, LO leakage and generated Doppler interferers. Thus, baseband filters are usually built to reject low frequencies, creating a void near DC, which is not critical in broadband operation, since it covers only a small portion of the signal bandwidth, but is often large enough to affect narrowband voice applications.

2.2.2 *Superheterodyne*

The superheterodyne receiver, shown in fig. 2.4, is the right choice for delivering high-performance narrowband and broadband receiver characteristics [23]. The RF signal is first downconverted into an intermediate frequency with the help of the first LO signal, often referred to as the "injection". The IF filter then protects against close-in spurious signals, and the signal is split into two channels, designated as I (in-phase) and Q (quadrature), each one being downconverted to baseband (zero center frequency).

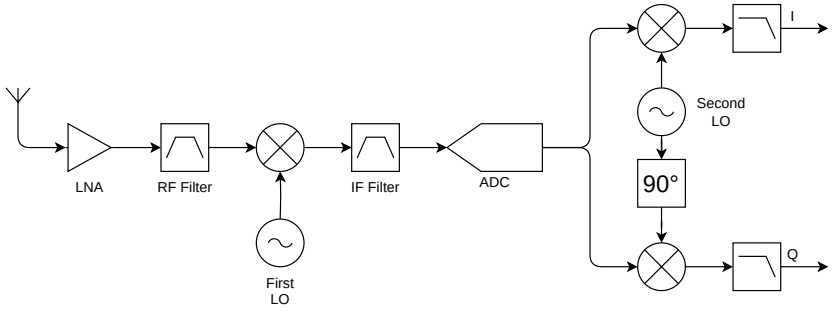


Figure 2.4: Superheterodyne receiver architecture.

The use of an intermediate frequency allows for a less complex circuitry in the IF signal path, as the design can be conceived to always operate on the same frequency (e.g., there is no need for IF filters with configurable center frequency), since the first LO can be adjusted such that the RF frequency of interest converts to the (fixed) IF frequency for which the receiver was designed. This reduces complexity without compromising the versatility of the receiver as a whole.

After the signal is downconverted to an IF, it is sampled, and handed over to a baseband processor in order to perform the final downconversion to complex baseband, detection and further processing on the received signal.

Some of the advantages and disadvantages of this architecture are presented below [23].

- Advantages
  - Due to the use of an intermediate frequency and a two-step conversion process, and since the quadrature baseband downconversion is done at a fixed IF frequency and in the digital domain, most low-frequency impairments (e.g., those introduced by a direct conversion receiver) can be mitigated, achieving an outstanding overall performance;
  - Only one analog-to-digital converter is needed, instead of two as in the direct conversion receiver.
- Disadvantages
  - Increased overall complexity;
  - Higher cost;
  - Large component count and occupied area;
  - Less suitable for integration, due to large size of lumped elements for lower operating frequencies (in the IF chain);
  - The signal domain converter needs to run at a higher sampling rate, since the signal is centered at IF and not at baseband.

2.2.3 *Direct RF sampling*

Recent technological advances have made possible the existence of higher speed data converters than before, which, in turn, allow the direct synthesization or capture RF signals without a conventional upconversion or downconversion with an analog radio chain [24] [25].

A direct RF sampling receiver represents the peak digitization available in the signal processing chain of a radio system nowadays. By feeding the output of the LNA directly into a high-speed analog-to-digital converter, one then takes digital control of the signal as soon as possible, allowing for greater flexibility. In fig. 2.5, as an example, a direct conversion topology is represented. However, since the data is already in the digital domain, which is completely reprogrammable, it is possible to implement whatever one wishes to. This full digitization comes, however, at the cost of very high data rates at the output of the RFADC, which not all digital processors can deal with.

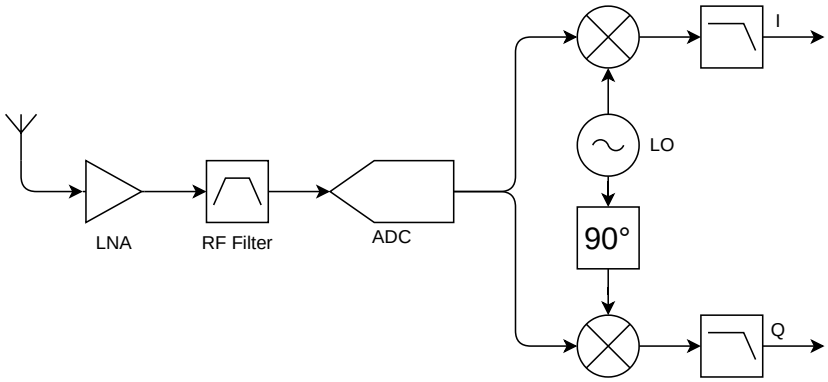


Figure 2.5: Direct RF Sampling receiver architecture.

Some of the advantages and disadvantages of this architecture are presented below [23].

- Advantages
  - Highly simplified analog RF frontend;
  - High configurability.
- Disadvantages
  - Although the bill of materials (BOM) is reduced compared to a more analog rich approach, the overall cost is still high;
  - Typically requires higher power to operate;
  - The signal domain converter needs to run at a very high sampling rate, and requires at least some part of the digital processing chain to happen at this sampling rate, until the signal is converted to baseband and decimated.

2.3 TRANSMITTER TOPOLOGIES

The most important transmitter architectures, addressed below, are the two-step conversion transmitter and the direct launch transmitter, which are useful for most broadband applications [23]. The more recent "DAC to Antenna" topology is also presented. All these architectures use an IQ modulator (either in software or hardware), and are suitable for both constant-envelope modulation methods, such as binary phase shift keying (BPSK) or quadrature phase shift keying (QPSK); and variable-envelope modulation schemes, such as quadrature amplitude modulation (QAM).

2.3.1 *Two-step conversion transmitter*

The two-step conversion transmitter, depicted in fig. 2.6, is analogous to the superheterodyne receiver topology described earlier. The I and Q baseband components are digitally upconverted into a digital intermediate frequency, which is then converted into the analog domain through a digital-to-analog converter. After the converter, a low-pass filter removes unwanted images of the signal created by the conversion process. The mixer then simply shifts the IF signal to the final RF transmission frequency. The band-pass filter (BPF) rejects unwanted image signal produced by the final mixer. The signal is then amplified by the power amplifier (PA) and transmitted through the antenna.

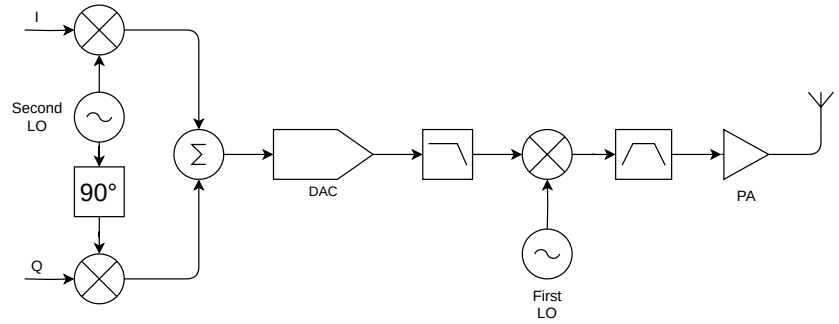


Figure 2.6: Two-step conversion transmitter architecture.

Since the design shares a lot of concepts with the superheterodyne receiver, the advantages and disadvantages are also pretty much the same.

2.3.2 *Direct launch transmitter*

The direct launch transmitter, represented in fig. 2.7, shares much of its architecture with the direct conversion receiver. Most low-cost and low-power broadband applications, such as Bluetooth and WiFi, utilize direct-launch architectures, as they are well designed for complementary metal-oxide semiconductor (CMOS) system-on-chip (SoC) integration [23].

The digital baseband I and Q components are directly converted into the analog domain by two DACs, low-pass filtered, and directly modulated onto the final RF frequency by a quadrature (image-reject) mixer. The signal is then amplified by the PA and transmitted through the antenna.

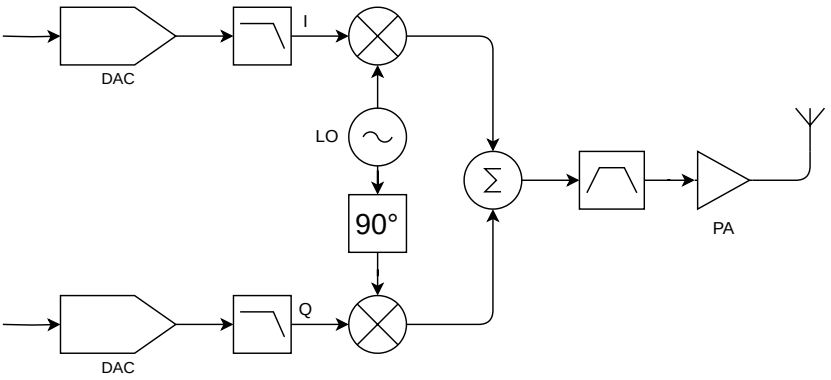


Figure 2.7: Direct launch transmitter architecture.

This topology shares the advantages and disadvantages with its receiver reciprocal, the direct conversion receiver.

2.3.3 *RFDACs - "DAC to Antenna"*

As with the direct RF sampling receiver, nowadays there also exist digital-to-analog converters capable of running at a very high sampling rate, in the giga-samples per second range, such as the one presented in [26], demonstrating that the reality of a truly software defined radio is closer than ever.

The "DAC to Antenna" transmitter concept aims to prove precisely that. Using an RF DAC running at 12 GSPS, one is capable of directly generating any signal up to 6 GHz, requiring very little analog inflexible circuitry. fig. 2.8 represents a digital implementation of a direct launch transmitter topology as an example. However, since everything is digitally implemented, one takes complete control over the signal generation, removing the limitations of fixed analog upconversion present in other transmitter architectures.

2.4 DIGITAL PROCESSING

The digital processor makes up the bulk of a software defined radio, and can be based on different types of commonly available processing hardware, such as general purpose CPUs, GPUs, FPGAs, digital signal processors (DSPs) or ASICs [12]. In this section, each one of the platforms is described and compared to each other.

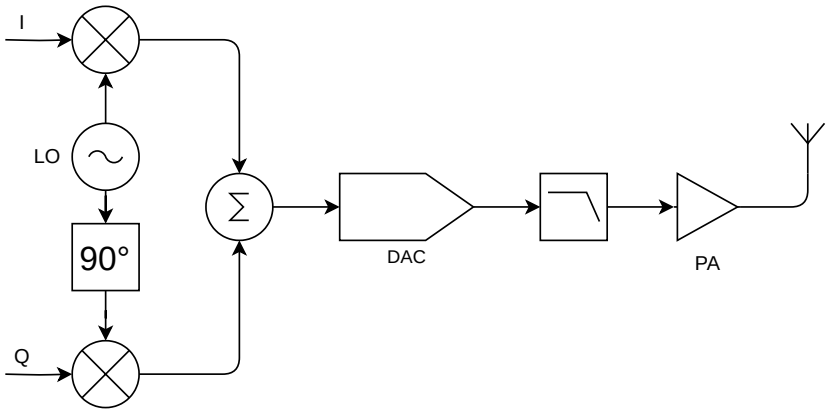


Figure 2.8: "DAC to Antenna" transmitter architecture.

2.4.1 *General purpose CPU*

General purpose CPUs are typically designed for personal computers or workstations. They are generally easier to develop applications for, using software libraries like [14] or [15]. However, those come at the cost of a higher energy consumption to achieve the same level of performance of other hardware [27].

2.4.2 *GPU*

Graphical processing units are, as the name implies, designed with graphics related algorithm acceleration in mind. There are, however, common properties between graphics processing and signal processing, not the least of which is the ability to speed-up execution by running algorithms in parallel. GPUs typically consist of multiple processing cores (e.g., NVIDIA's compute unified device architecture (CUDA) cores), which can be configured to run signal processing algorithms, as demonstrated by [28]. There exist software abstraction layers, such as [29] and [30], that allow easier integration of GPU accelerated signal processing algorithms in already existing non-accelerated applications.

2.4.3 *FPGA*

FPGAs are semiconductor devices which consist of regular logic structures (e.g., a look-up table (LUT) and multiplexer, flip-flops) and interconnect, that are reconfigurable to represent any kind of logic circuit, from single gates (e.g., AND, OR,

XOR), to highly complex logic circuits. The advantages of the architecture of an FPGA allow the designs to perform various computational operations in parallel. Parallelism enables substantial data throughput at relatively low clock rates [31]. Nowadays, with techniques such as high-level synthesis (HLS), programming an FPGA is becoming more accessible to software engineers and everyone who is used to write code in programming languages with high levels of abstraction [32]. However, in order to further optimize the implemented logic circuit, it is still needed to have some knowledge of the underlying hardware architecture.

2.4.4 *DSP*

A DSP is a special type of microprocessor to process digital signals [33]. DSPs, much like general purpose CPUs, are capable of implementing and executing complex arithmetic and logic tasks [34]. DSPs, however, due to their architectural structure (i.e., reduced instruction-set computing (RISC) architecture, single instruction-multiple data (SIMD) extensions), tend to handle arithmetic operations, especially multiplications, quicker. Since DSPs are capable of delivering high performance with lower power, they are better candidates for SDR deployment, compared to general purpose CPUs [35].

2.4.5 *ASIC*

Presented with the original concept of an application specific integrated circuit, one might tend to question its applicability in a software defined radio environment, where reconfigurability and flexibility are key. Digital ASICs can implement a logic circuit, described in the same manner as it would be for an FPGA. The energy efficiency is typically greater than that of an FPGA, and it can usually run at higher frequencies. Despite not being able to alter the implemented circuit of ASICs, they can be of great benefit, especially to the power consumption of the system [36], without being a major hit in the overall flexibility, if employed in the correct places. Processing chain components that are common to a wide range of applications, such as digital downconverters, digital upconverters, may be implemented in an ASIC. Regular structures, such as FIR filters, can also be implemented as an ASIC, given that the computation required is not related at all with the frequency response of the filter, i.e., the multiply-accumulate operation is common to all FIR filters, but the shape of the frequency response is defined by its impulse response.

2.5 STATE-OF-THE-ART REVIEW

Nowadays, building a test software defined radio platform for mmWave frequencies is highly accessible, given the existence of integrated up and downconverters, that can be combined with a lower frequency SDR to reach mmWave bands. Examples of such converters are the ADMV1013 and ADMV1014, recently launched by Analog Devices. These converters are highly flexible, allowing an LO input frequency from 5.4 GHz to 10.25 GHz, which is internally multiplied to achieve frequencies in the mmWave band. Besides allowing baseband I/Q data input from DC up to 6 GHz, they can also perform frequency translation on an already modulated carrier, from 0.8 GHz to 6 GHz. The ADMV1013 and ADMV1014 offer exceptional performance attributes that simplify the design and implementation of compact 5G mmWave platforms. These advanced devices effectively cover the widely used 28 GHz and 39 GHz frequency bands, making them suitable for both backhaul and fronthaul applications. Additionally, their capabilities extend to a wide range of ultrawide bandwidth transmitter and receiver applications, further enhancing their versatility and usability in diverse mmWave communication scenarios, and making them highly suitable candidates for use in a mmWave SDR RF frontend [37].

The development of very fast data converters (DACs and ADCs) in academia has demonstrated remarkable progress in pushing the boundaries of sampling rates, available bandwidth and data acquisition speeds. In [38], the authors were able to double what was state-of-the-art at the time, presenting an ADC with 60 GHz of available bandwidth. However, despite the promising academic results, it is important to acknowledge that the commercialization and mass production of such high-speed converters still faces significant challenges. The transition from research prototypes to market-ready products requires careful consideration of factors such as reliability, power efficiency, cost-effectiveness, and scalability.

Currently, Analog Devices and Texas Instruments lead the commercial high-speed ADC market. At the International Microwave Symposium in Boston, in 2019, Texas Instruments introduced the ADC12DJ5200RF, a 12-bit, 10.4 GSPS RF-sampling ADC, with a usable bandwidth up to 8 GHz. Its performance was also first demonstrated at this event. Analog Devices, however, leads the high-speed DAC segment, with the AD917x family, achieving up to 12.6 GSPS, with a spurious-free dynamic range (SFDR) better than -80 dBc at 1.84 GHz.

In the commercial off-the-shelf (COTS) department, there is Per Vices Corporation [39], a leading company in the field of software defined radios that has

made significant contributions to the advancement and commercialization of SDR technology. Their SDR platforms offer a wide range of capabilities and are designed to meet the demanding requirements of modern wireless communication systems and research applications. Currently listed on their website, there are three readily available SDR models: *Cyan*, *Chestnut* and *Crimson TNG*. Relevant for this analysis is the *Cyan* SDR, which offers a tuning range up to 18 GHz. Also pertinent is their custom SDR configurator, which allows requesting the design of a software defined radio platform which can tune up to 86 GHz, with an instantaneous TX and RX bandwidth of 2.5 GHz. The implementation used to achieve such performance levels, is, however, not disclosed.

Despite the remarkable evolution in high-speed converters, it is quite clear that, in order to reliably reach frequencies in the mmWave range in the present day, a direct or multiple conversion approach is still required.

In light of the significant advancements witnessed in high-speed converters, it is indisputably evident that attaining frequencies within the mmWave bands, for the time being, necessitates the adoption of a direct or multiple (superheterodyne) conversion approach to ensure consistent and reliable performance. These facilitate the efficient translation of signals between the mmWave spectrum and lower frequency bands, enabling seamless integration and seamless transmission of data in current and near-future communication systems. Continued improvements in high-speed converters, coupled with the refinement of direct and multiple conversion approaches, hold the key to unlocking the full potential of the mmWave radio spectrum and propelling the next generation of wireless communication technologies to unprecedented heights.

2.6 SYSTEM ARCHITECTURE

The objective of this work is to explore the architecture and potential implementations of SDRs, and to develop, implement, and test a software defined radio capable of operating across a wide frequency range, from several megahertz up to tens of gigahertz, extending into the millimeter-wave frequency region, as stated in section 1.3.

An SDR covering a frequency range from several megahertz up to the mmWave frequency region should be based on frequency conversion techniques rather than direct sampling for several important reasons. Firstly, the sampling rate required for direct sampling of such a wide frequency range would be prohibitively high.

According to the Nyquist-Shannon sampling theorem, the sampling rate must be at least twice the highest frequency component to accurately represent the signal. For a 40 GHz signal, this would require a sampling rate of at least 80 GHz, which is far beyond the capabilities of current commercially available analog-to-digital converters and digital-to-analog converters, as was shown in section 2.5, although they can be found at the academic research level. Secondly, frequency conversion techniques, such as heterodyning (or even a homodyne), allow for more efficient and flexible signal processing. By using mixers to shift high-frequency signals down to a lower intermediate frequency (or baseband), the SDR can work with more manageable frequencies. This approach enables the use of lower-speed, higher-resolution ADCs and reduces the computational requirements for digital signal processing. Furthermore, frequency conversion techniques provide better noise performance and dynamic range when compared to direct sampling at extremely high frequencies. As the frequency increases, the noise figure of ADCs tends to degrade, limiting the receiver's sensitivity. By downconverting the signal before digitization, the SDR can maintain better signal quality and achieve a higher signal-to-noise ratio. Finally, an SDR based on frequency conversion techniques rather than direct sampling present significant advantages in managing spurious emissions and improving SFDR. Direct RF sampling at such high frequencies would require extremely high sampling rates (tens of giga samples per second), leading to increased quantization errors and timing jitter in the ADCs and DACs, which in turn generate unwanted spurious signals. These spurious emissions can interfere with signals of interest, compromising data integrity and potentially masking weak signals, reducing sensitivity. Therefore the SDR architecture will be based on heterodyne conversion techniques.

To cover such a wide frequency range and optimize the SDR performance, the SDR was split into two subsystems: the SDR base platform and the expansion module, as illustrated in fig. 2.9 by the high-level block diagram of the SDR system. This design approach ensures that the SDR system is useful for as many application scenarios as possible, since it is architectured in an expansible way, leaving, where possible, room for future feature implementations.

The SDR base platform should house a base RF transceiver capable of handling signals with frequencies from at least 100 MHz to 6 GHz. This RF transceiver should employ a hybrid transmitter/receiver topology, employing a direct conversion architecture and performing signal domain conversion (analog-to-digital and digital-to-analog) directly at baseband, appropriate for RF signal bandwidths of at least 50 MHz, instead of RF sampling units (RFADCs and RFDAC). With the proposed

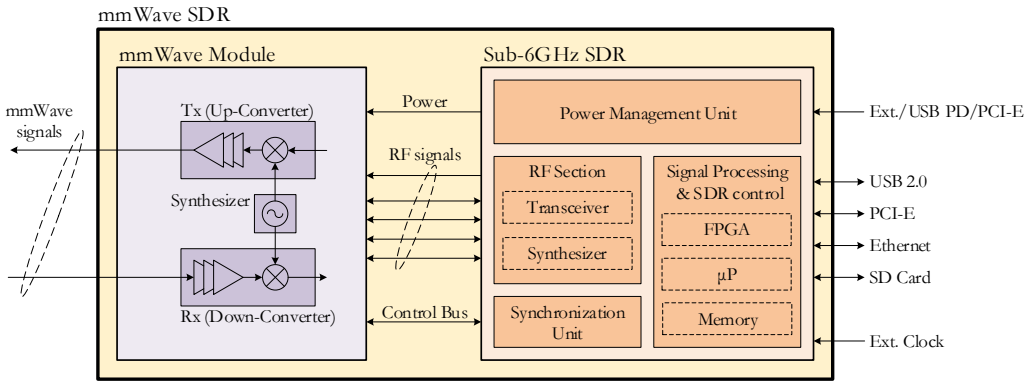


Figure 2.9: SDR architecture block diagram.

architecture, RFADCs and DACs could be utilized in place of the conventional transceiver, if it weren't for their high prices, which is presently prohibitive. The SDR base platform should support 2x2 multiple-input multiple-output (MIMO) ports, with each TX/RX channel featuring two digitally selectable input/output signal paths (for different frequency bands, for example). The output power should reach up to +10 dBm for frequencies up to 6 GHz. Additionally, the base platform should include an on-board frequency synthesizer, with the purpose of generating the local oscillator for expansion modules (daughter cards). This synthesizer may also be usable for other purposes, and should operate from several MHz up to at least 18 GHz.

All digital signal processing and SDR control occurs within the SDR base platform, including on-board processing in hardware (e.g., FPGA) and/or software processing on a central processing unit (e.g., ARM Cortex core), such as modulation, demodulation, filtering, and block configuration, control, and monitoring. This block serves as the "brain" of the radio. Data offloading should be performed via PCIe at up to 8 Gbps (Two-lane PCIe Gen2 link).

The synchronization unit generates clocks for all the functional blocks, assuring the required synchronization between clock signals, and should be able to support at least a clock input and output to allow coherent (synchronized) operation of connected devices on a system where the SDR may be employed.

The SDR base platform should be capable of operating via standard DC input from 9 V to 24 V, universal serial bus power delivery (USB PD), or the power rail provided by a PCIe slot. It should comprise a power management unit (PMU), responsible for generating, controlling, sequencing and monitoring all the power supply rails required by the internal functional blocks of the radio (base platform

and expansion cards). The power management unit of the SDR base platform must be over-provisioned to accommodate various daughter cards.

The expansion module enhances the flexibility and modularity of the SDR system by providing accommodation for various and different (specific applications) daughter boards, such as satellite ground stations, fixed microwave or mmWave links, higher power transmitting amplifiers, lower noise receiving frontends, among many others. A daughter board can, when connected to the base platform, function as an up and down frequency converter for the SDR base platform, effectively extending its frequency range. This modular approach allows the SDR system to adapt to diverse operational requirements and frequency bands beyond the capabilities of the base platform alone.

The proposed architecture for the SDR system offers flexibility in various aspects, such as frequency bands, bandwidths, sensitivities, and transmitted signal powers, by accepting different frontend modules. Moreover, the SDR can be controlled via a personal computer (PC) through the PCIe bus, or by an attached single board computer (SBC), exposing an Ethernet interface for data offloading. This topology makes the SDR system highly versatile for a range of applications. The technical specifications were also defined to cover the widest possible range of use cases.

To effectively cover the mmWave band, it is also necessary to develop an expansion module compatible with the base platform. The mmWave module consists of a transmitter and receiver chain that translate mmWave signals to and from lower frequencies, respectively. This specialized card should encompass the 24 GHz to 40 GHz frequency range, capable of both transmitting (TX port) and receiving (RX port) mmWave signals. The board should be designed to output signals with power levels up to 0 dBm. The expansion board will interface with the SDR base platform, handling intermediate frequency signals centered at 3 GHz for both reception and transmission. The IF channels (TX and RX ports) should, at minimum, match the bandwidth capabilities of the SDR base platform's RF channels, which is 50 MHz or greater. Implementing higher bandwidths would enhance the versatility of the mmWave daughter board, enabling its use in systems beyond the SDR base platform. To maximize flexibility, the board should be designed with the capability to operate using either an internal local oscillator or an external one, such as the on-board synthesizer integrated into the SDR base platform. The mmWave card topology should support full-duplex operation of the transmitter and receiver paths, with the additional feature of allowing different central frequencies for each path (i.e., frequency-division duplex (FDD) operatoin). The mmWave module should

be designed to receive power from the PMU of the SDR base platform, as well as communicating with it for configuration, control and monitoring.

3

SDR BASE PLATFORM CIRCUIT DEVELOPMENT

This chapter addresses the process of designing the PCB for the base SDR platform, which works up to 6 GHz. It is divided into three sections, where the first comprises a comprehensive description of the platform architectures, using a functional block diagram. The second section extensively addresses the circuit design process, from the selection of components to their interconnections. Finally, the PCB layout process is clarified in detail, including the challenges faced during this process, and the methodology used to overcome them.

3.1 PLATFORM ARCHITECTURE

This section describes, in detail, the architecture of the SDR base platform. To ensure the SDR usability in as many application scenarios as possible, it is architectured in an expansible way, leaving, where possible, room for future feature implementations, as can be seen by the top level block diagram of fig. 3.1. The technical specifications were also defined to cover the widest possible range of use cases.

The synchronization unit generates clocks for the digital logic and reference signals for various functional blocks (i.e., transceiver and frequency synthesizers), assuring the required synchronization between clock domains. It is comprised by an internal clock source and a highly programmable clock generator integrated circuit (IC). The synchronization unit allows the SDR user to input an external clock source (e.g., from a GPS disciplined oscillator (GPSDO)) as well as providing the user with a clock output for synchronizing external devices. The clock tree of the platform should be as programmable as possible, allowing, for example, the user to select which clock source to use (internal or external) for a certain clock output. It should also achieve a determined output clock frequency with excellent precision.

The signal processing and SDR control block is where all the digital signal processing occurs, such as modulation, demodulation, filtering and block configuration, control, and monitoring. This block serves as the "brain" of the radio and comprises

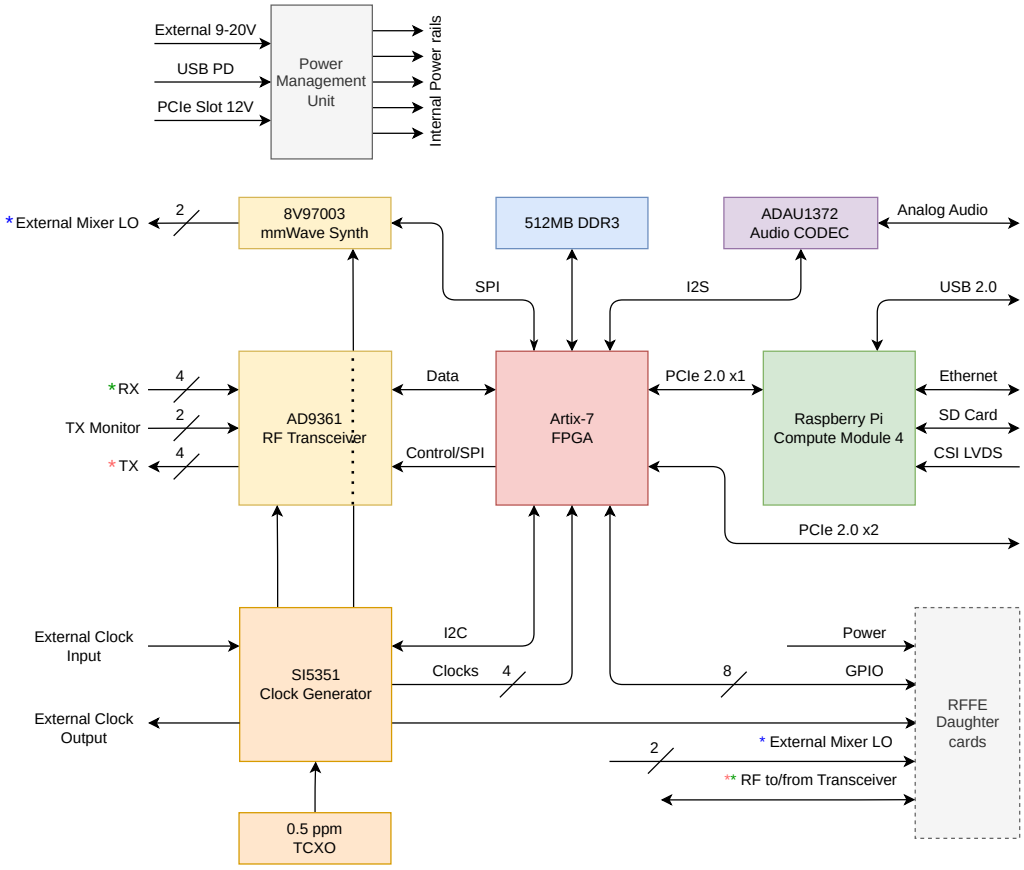


Figure 3.1: Base SDR platform block diagram.

a FPGA, which communicates via PCIe with an on-board Raspberry Pi compute module 4 (CM4) or with an external Root Complex (host) through an edge connector. The FPGA should have more than enough logic resources for accelerating and even completely implementing DSP algorithms, although it can be programmed to serve simply as a translation layer between the RF transceiver and the host controller. A block of DDR3 random access memory (RAM) is also included and directly connected to the FPGA, which should further increase the flexibility of the SDR when operating standalone. The integrated Raspberry Pi CM4 increases flexibility by allowing digital signal processing to happen at the software level, instead of at firmware or hardware level as would be the case of the FPGA.

The RF section comprises an agile RF transceiver which performs digital filtering, signal domain conversion, frequency conversion (translation) and signal amplification. The agile transceiver operates up to 6 GHz, but its output signals can be routed to expansion cards (modules) which will further translate the signal into other frequency bands. The base platform also includes a frequency synthesizer IC capable of generating signals with frequency up to 18 GHz. These are meant to serve as LO for expansion cards, but can be used for other diverse purposes.

The base platform is architected such that further expansion is possible, and this point should always be taken into consideration during the design phase. In the future, additional expansion cards (other than the mmWave module, for example) could be developed, either to extend the frequency span to higher or lower frequencies, and/or to provide higher output power, among other desirable features. The expansion modules will have access to the RF transceiver inputs and outputs, the frequency synthesizer outputs, various power rails with different voltages and current supply capabilities, general purpose digital I/O to/from the FPGA and a clock signal from the synchronization unit.

Although not an essential component for the principal SDR operation goals, an audio coder / decoder (CODEC) is also included in the design, as part of the flexible architecture objective. This block allows easier interfacing with analog audio peripherals.

3.2 FUNCTIONAL BLOCK DESIGN

In order to determine the needed components and start selecting suitable parts, a highly simplified block diagram of the system was drawn, including just the main sections of the SDR and the interconnections that may exist between them. This diagram is represented in fig. 3.1, which is a visual representation of the architecture briefly explained in section 2.6.

3.2.1 *Signal processing core*

The core of the system is an FPGA, given its high performance and flexibility, as explained in section 2.4. When it comes to FPGA manufacturers, two names arise immediately: Altera (now Intel) and Xilinx (now AMD). Although there are many more manufacturers of FPGA ICs, Altera and Xilinx definitely offer the largest ecosystem of products, tools and support. For that reason, and taking into account previous experience with Xilinx products, a Xilinx 7-series FPGA was chosen, namely the XC7A100T in an FGG484 package. This part contains 101440 logic cells, 240 DSP slices and 4860 kb of block RAM. The more than 280 general purpose input/output lines are spread across four banks with each one supporting operation on a different I/O standard (e.g., LVTTL, LVCMOS, LVPECL, LVDS, etc...). There are drop-in replacement products in the same family which are pin-compatible with this part, offering different amounts of logic resources, and even parts designer for

lower power operation. This means that once the design is finalized, a cheaper part can be installed, shall the design work correctly on said part.

Accompanying the FPGA, a single 512 MB DDR3 memory is used for general purpose data storage, further increasing the flexibility of the SDR by offering the ability to execute processing functions that require more memory than the FPGA block RAM can offer.

The FPGA communicates with the application processor through PCIe. The processor can either be an integrated Raspberry Pi CM4 or any other PC with an available PCIe slot. The CM4 is a system-on-module (SoM) consisting mainly of a Broadcom BCM2711 SoC, RAM and, optionally, non-volatile storage. The module provides a lot of connectivity through two 100-pin board-to-board connectors, such as two MIPI camera serial interfaces (CSIs), two MIPI display serial interfaces (DSIs) and two high-definition multimedia interfaces (HDMIs) which can operate at 4K resolution and 60 frames per second. Additionally, various low-speed communication buses are also provided among with a single lane of PCIe which can operate at 5 GT/s. The use of a socketed SoM directly contributes to future-proofing the SDR platform, as an upgraded version of the module can easily be swapped-in, instantly providing higher computing power to the SDR.

3.2.2 *RF transceiver*

In regards to highly integrated ultra-wideband RF transceivers, Analog Devices seems to lead the market. Many COTS software defined radio equipment seems to utilize transceivers from the AD936x family. The Lime Microsystems LMS7002M is also known in the SDR COTS market, however, in spite supporting a wider bandwidth of 120 MHz, its frequency range is limited to 3.8 GHz, while the AD9361 can reach 6 GHz with a sample rate of up to 61.44 MSPS, a typical receiver noise figure of 2 dB, a TX noise floor better than -157 dBm/Hz, while also offering numerous digital signal processing and monitoring functions. In addition, the Analog Devices parts seem to be better documented and supported, both by the manufacturer and the community. As such, the AD9361 RF transceiver seems to suite the needs of this project, covering frequencies from 70 MHz (47 MHz TX) all the way up to 6 GHz at a maximum instantaneous bandwidth of 56 MHz. As can be seen by the device internal block diagram in fig. 3.2, the transceiver comprises three RX inputs and two TX outputs per channel, totaling six RX and four TX independent connections. As per the SDR block diagram, fig. 3.1, all the four TX

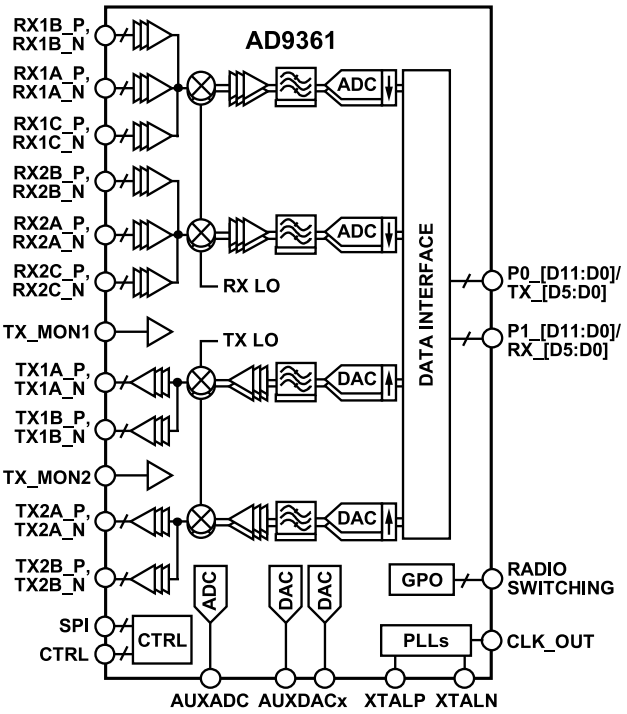


Figure 3.2: AD9361 functional block diagram [40].

outputs are exposed to the user. The A output of each channel is exposed directly to the user, while the B output is used to feed an ADL5601 amplifier, whose output is then available via an SMA connector. The ADL5601 is a general purpose gain block from Analog Devices which operates at up to 4 GHz, and is used in the SDR to provide an output with a higher power level than is normally provided by the transceiver. The ADL5601 is chosen because of it's package footprint, which is compatible with many other general purpose amplifiers, including parts from other vendors, adding to the versatility and flexibility of the SDR platform.

On the receiving side, the base SDR only exposes four inputs. The A input of each receiver is exposed directly, as the A channel of the transmitters. The C input of both receivers is also exposed to the user, however, it has the added capability of supplying DC power to an external LNA or any other active device, through two digitally controllable wideband Bias-Ts, implemented externally to the transceiver. The B input of the receivers is not exposed, and its connection pads are shorted to ground on the board, as per the manufacturer recommendation for unused RF inputs.

The general purpose outputs (GPOs) of the transceiver, depicted in fig. 3.2, are used to control the TX amplifiers and RX Bias-Ts. These digital outputs can be directly associated with the transceiver signal processing chain control logic, such as

to only supply power the relevant elements when needed (e.g., only power the TX amplifiers when actively transmitting, or the RX Bias-Ts when additional gain is needed from an external LNA). The crystal inputs depicted in fig. 3.2 (XTALP/N) are used in this design as external clock inputs, connected to the synchronization unit (fig. 3.1), as this is a configuration supported by Analog Devices.

3.2.3 *mmWave synthesizer*

As shown in the block diagram of fig. 3.1, a synthesizer capable of generating high-frequency tones is included in the design. The purpose of this block is to generate local oscillator signals for the expansion boards. Several parts were considered, mainly from Analog Devices, such as the ADF4371. This part covers a large frequency range, from 62.5 MHz to 32 GHz, however, it does not utilize a single output for the entire frequency range, as can be seen in the functional diagram of fig. 3.3, meaning plenty of additional components are needed shall an applications' frequency range cross the boundary between two of the ADF4371 outputs [41].

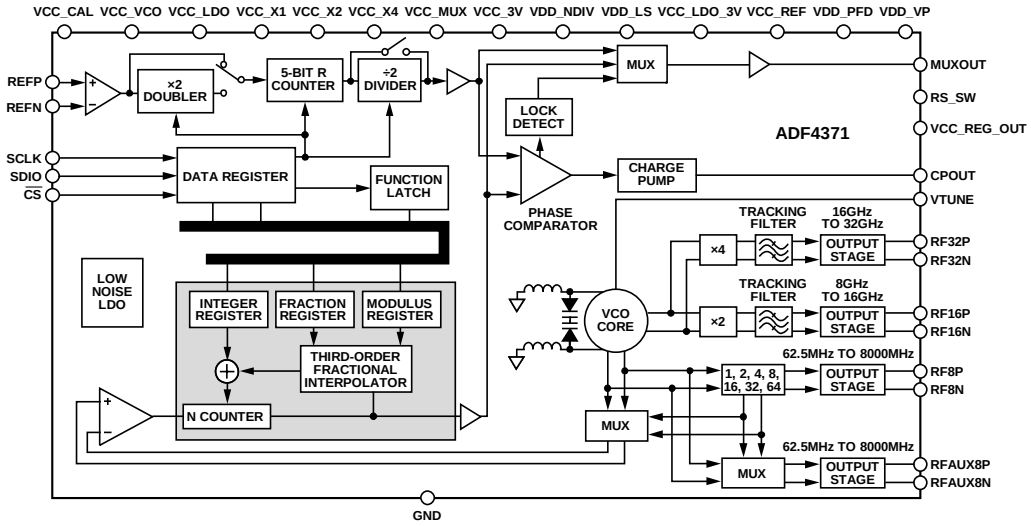


Figure 3.3: ADF4371 functional block diagram [41].

The Renesas Electronics 8V97003 covers a frequency band from 171.875 MHz to 18 GHz, which is significantly lower than the ADF4371, however, besides accomplishing a slightly lower phase noise floor of -236 dBc/Hz, it also outputs the entire range through a single pair of differential outputs, as shown in its internal diagram in fig. 3.4. Given that this will be the signal with the highest frequency of the entire design, it will practically dictate the PCB materials that can be used. Achieving a PCB design capable of reliably operating at or near 32 GHz requires careful

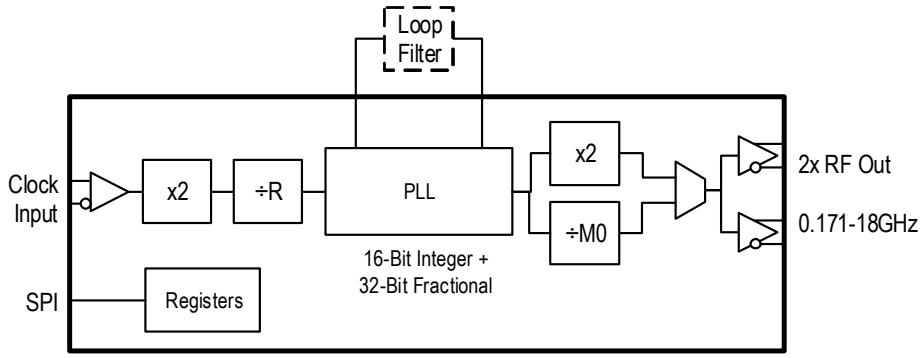


Figure 3.4: 8V97003 simplified block diagram [42].

material selection, which quickly raises the cost of PCB manufacturing, and even at 18 GHz proves to be quite challenging. Because of this, a upper-end frequency of 18 GHz is decided to be enough for the purpose of this platform, considering the lower power that can be extracted from the synthesizer at 32 GHz when utilizing a typical FR-4 dielectric PCB. A more suitable substrate could, of course, be utilized, such as Rogers 4350B or 4003C. However, it would significantly increase the cost of the PCB, as mentioned before, for the sole purpose of generating a 32 GHz tone with usable power levels. The decision to not support such high frequencies is also justified by the fact that many converter architectures that operate on those mmWave bands only require an input LO signal that is a divided-down version of the final RF carrier frequency [43, 44].

In addition to the excellent voltage controlled oscillator (VCO) phase noise performance, the 8V97003 also offers an input reference multiplier, as can be seen in fig. 3.4, which allows clocking the phase-frequency detector (PFD) at a higher frequency even when the actual input reference frequency is lower. A few relevant benefits of operating the PFD at higher frequencies are improved phase detection resolution, faster lock times and reduced spurious noise [45]. For those reasons, the 8V97003 synthesizer IC was ultimately chosen.

As can be seen in fig. 3.4, the PLL loop filter is expected to be implemented externally to the IC, using discrete components. The loop filter is a critical component that shapes the overall performance of the PLL by controlling stability, noise rejection, and transient response. It determines how the PLL responds to phase errors and affects key parameters such as lock time, jitter performance, and reference spurs [46]. The implemented 3^{rd} order passive loop-filter topology is depicted in the schematic of fig. 3.5, and was designed with the considerations from various Renesas Electronics provided application notes [47, 48]. The PLL loop bandwidth,

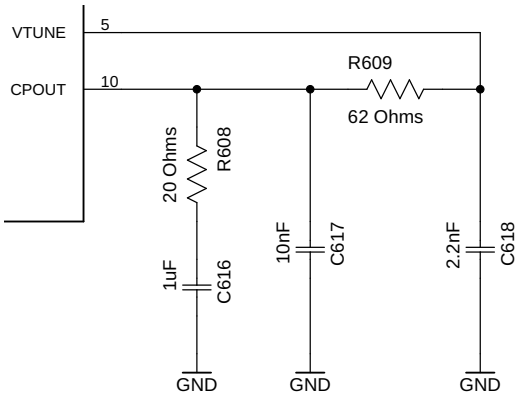


Figure 3.5: mmWave synthesizer loop filter schematic.

f_c, which influences not only the lock time but also the phase noise profile of the synthesized signal, can be expressed by eq. (3.1),

$$f_c = \frac{R_s \cdot I_{CP} \cdot K_{VCO}}{N \cdot 2\pi}, \tag{3.1}$$

where R_s is R_{608} in fig. 3.5, I_{CP} is the charge pump current, K_{VCO} is the the VCO gain and N is the feedback divider. As can be seen, for a given PFD and output frequency pair, the only remaining variable which affects the loop bandwidth is the charge pump current, I_{CP}. The 8V97003 IC offers a digitally controllable charge pump, yielding a finer control over the loop bandwidth of the PLL. This feature is exploited to achieve an automatic loop tuning when configuring the synthesizer. At the schematic level, the value of R_s (R_{608}) was picked such that with a variation of the charge pump current, a wide variety of loop bandwidth configurations was achievable, over a reasonable range of operating PFD frequencies, as per the datasheet [42]. The whole loop filter, however, can be re-designed shall a specific application of the SDR require so.

The remaining loop filter components also play crucial roles in the synthesizer performance, as they directly determine the zero frequency, f_z, the first pole frequency, f_p, and the second pole frequency, f_{p2}, as per the following equations:

$$f_z = \frac{1}{2\pi R_s C_s}, \tag{3.2}$$

where C_s is C_{616} in fig. 3.5,

$$f_p = \frac{1}{2\pi R_s C_p}, \tag{3.3}$$

where C_p is C_{617} in fig. 3.5, and

$$f_{p2} \approx \frac{1}{2\pi R_3 C_3}, \tag{3.4}$$

where R_3 and C_3 are R_{609} and C_{618}, respectively, in fig. 3.5. As a rule of thumb, to keep the PLL operating in a stable region, the relationship $f_z < f_c < f_p < f_{p2}$ should be respected [47]. As the only variable component is f_c, the aforementioned control algorithm is crucial to ensure stability.

Regarding the output ports of the synthesizer, the datasheet mentions the requirement for external biasing, due to the open-collector nature of the complementary outputs. This biasing can be implemented using resistive or reactive elements, but the resistive topology is employed here due to the operating frequency limitations introduced by reactive terminations. Radio frequency resistors from Vishay are used to bias the outputs of the synthesizer, as depicted by fig. 3.6. To separate the DC

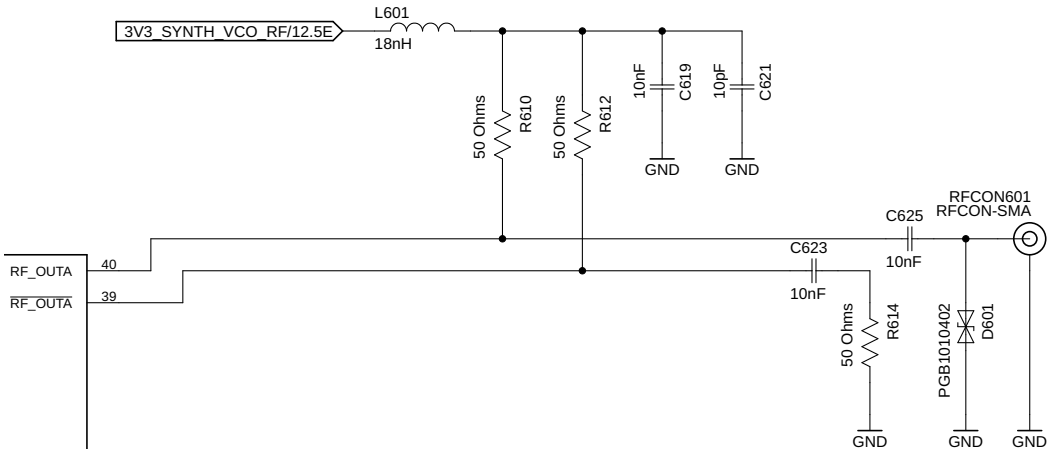


Figure 3.6: mmWave synthesizer output network schematic.

level of the biased synthesizer outputs from the externally connected circuit, a DC-block capacitor is used. A wideband 520L series DC-block from American Technical Ceramics is employed here, as it is specified to introduce minimal attenuation up to 16 GHz, and recommended by the synthesizer IC manufacturer, Renesas Electronics. As a single ended output is desired, the complementary ports of both channels are terminated with another RF load resistor.

3.2.4 *Clock management*

The Skyworks Si5351 clock generator is responsible for synthesizing various clock and synchronization signals required by certain system blocks, mainly the FPGA,

the RF transceiver, mmWave synthesizer and expansion cards. This part, which is functionally depicted in fig. 3.7, offers eight independent clock outputs which can produce non-integer multiples of up to two different reference inputs. It is a compact solution to distribute multiple reference signals, compared to utilizing a single crystal per block. The two clock inputs are a key advantage in this application, allowing one (the XA/XB) input to be used by an on-board 0.5 ppm temperature-compensated crystal oscillator (TCXO) while exposing the second (the CLKIN) input to the user, which can provide their own master clock if required. The system is then able to either synchronize completely or selectively to this user-provided clock.

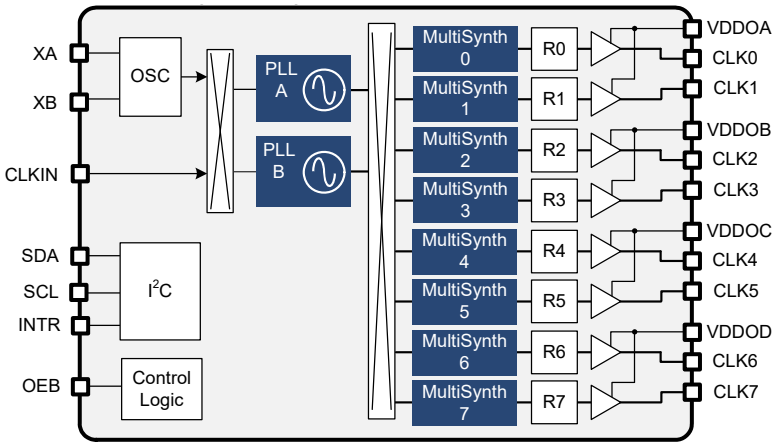


Figure 3.7: Si5351C simplified block diagram [49].

Each MultiSynth depicted in fig. 3.7 divides the source clock by a configurable fractional ratio and there's an output stage for each of the MultiSynth divided clocks. The source clock for the MultiSynth is one of two PLLs, which comprise a programmable fractional feedback divider. The source clock for each PLL can be originated on the TCXO or on an external clock.

3.2.5 *Power management*

The SDR platform can be powered externally through a two wire standard DC power source, via the 12 V rail of the PCIe edge connector or via USB PD. The PMU is responsible for generating, controlling, sequencing and monitoring all the power supply rails required by the internal functional blocks of the radio. It comprises a power management controller (PMC) microcontroller, which, among other tasks, handles USB PD communication and negotiation, selects the best power source to use and correctly configures the voltage regulators to produce all the voltage rails required by the system, while also sequencing them properly. Figure 3.8 depicts the

SDR power supply tree, including the three aforementioned possible power inputs, all the voltage regulators, voltage levels and maximum current supported by each rail. This figure also contains the PMC and its connections to the power tree. This detailed diagram is further explained in this section.

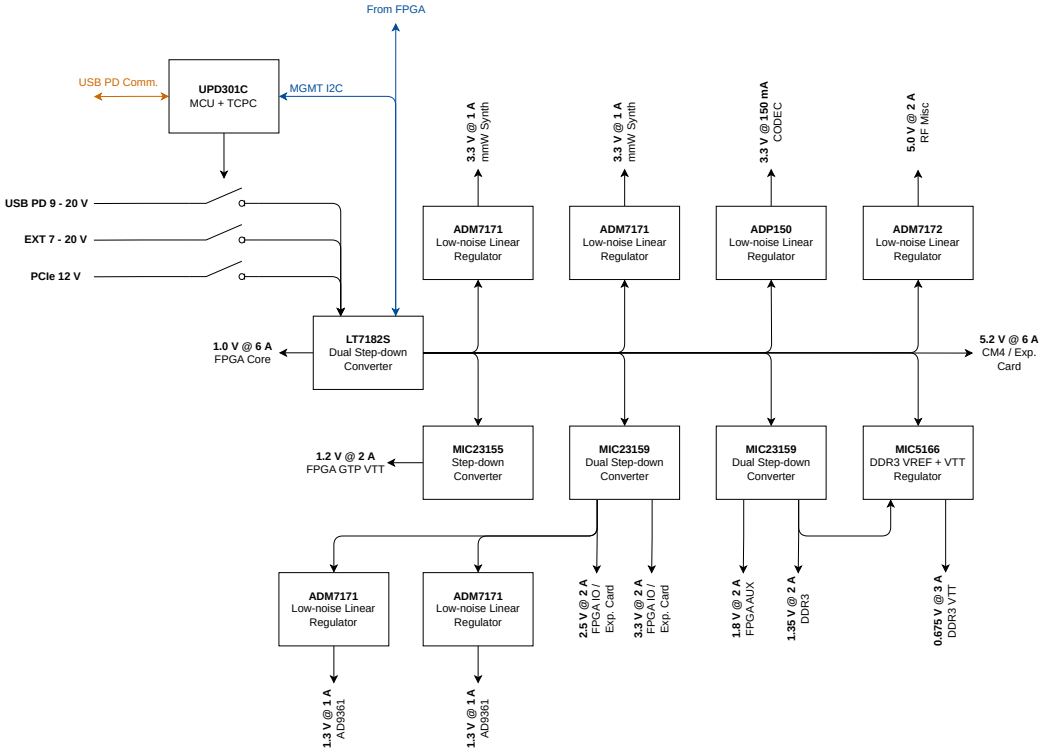


Figure 3.8: Base SDR platform power management unit diagram.

There are several voltage rails needed to power all the working sections of the SDR, each having different maximum power and noise constraints that should not be overlooked. This, associated with the previously mentioned design goal of supporting three different power sources for the SDR, means that the power management unit is a quite complex system on its own.

When it comes to the architecture of the PMU, it is important to understand the two fundamental voltage regulator families: linear regulators and switching converters. Switching regulators and linear regulators each have their own strengths and weaknesses, making them suitable for different applications. Switching regulators are highly efficient, often exceeding 80-90%, as they use inductors, capacitors, and switching elements to convert voltage with minimal power dissipation. However, they introduce high-frequency noise and electromagnetic interference (EMI), which can be problematic for sensitive analog or RF circuits which comprise the SDR. On the other hand, linear regulators, while offering excellent noise performance and fast transient response, suffer from poor efficiency, especially when the voltage

drop between their input and output is significant, resulting in excessive energy loss from heat dissipation. To achieve an optimal balance between the two, a common approach is to use switching regulators as pre-regulators, setting their output voltage slightly higher than the required final voltage, followed by linear low dropout (LDO) regulators to provide clean, stable power to noise-sensitive components. This hybrid approach maximizes efficiency while ensuring low noise, as the switching regulator handles most of the power conversion efficiently, and the LDO removes residual ripple and switching noise, delivering a quiet supply voltage ideal for precision analog, RF, and low-noise digital applications [50]. This hybrid architecture can clearly be observed in fig. 3.8.

The UPD301C is a multi-die integrated circuit designed for USB PD applications. It contains not only a USB PD controller, but also a general purpose ARM Cortex-M0+ micro-controller unit (MCU). The architecture of the IC is quite interesting, since the MCU is programmable independently of the controller, and thus offers a lot of flexibility when it comes to the specific application scenario. In this use case, it is used not only to manage the universal serial bus (USB) port power, but also to manage the whole system power tree.

Correct sequencing of power rails in systems with multiple supply rails and ICs that require different voltages is crucial for reliable operation and long-term stability. The order in which different power rails are turned on and off can affect functionality, performance, and even the longevity of components [51]. Many modern ICs have internal protection diodes between power rails and I/O pins, and if the rails are powered in the wrong sequence, unwanted current paths may form, potentially damaging the IC. Some chips, especially FPGAs and processors, have separate core and I/O voltages, and if, for example, the I/O rail is powered before the core, it can result in latch-up, where excessive current flows through unintended paths, permanently damaging the device. In this SDR platform, the power sequence is guaranteed by the correct chaining of regulators, where the power-good indication signal of the previous regulator is connected to the enable signal of the next one.

The UPD301C PMC is the first part of the SDR circuit to power-on, and is always powered, no matter what the SDR is doing. At boot, it first decides which power source is to be used (USB PD, PCIe or external input), and connects it to the system pre-regulator, allowing the rest of the system to power up following the sequence depicted in fig. 3.9, ensured by the aforementioned chain connection topology. During operation, the PMC communicates with the FPGA, which monitors and manages the system power.

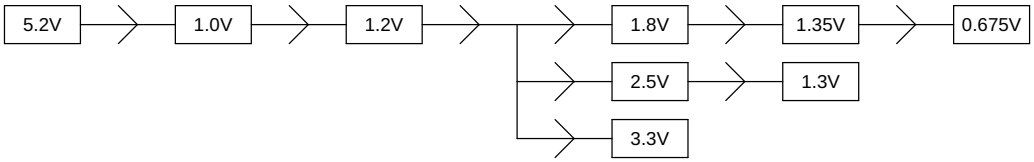


Figure 3.9: Simplified PMU voltage rail power-up sequence.

The LT7182S is a feature-rich dual step-down switching regulator, with support for configuration and telemetry via the power management bus (PMBus). It is used in this system as a pre-regulator, due to its high input voltage tolerance of up to 20 V and high output current capability of up to 6 A per channel, continuously, or, at most, one of the channels may provide 8 A. The first output channel of the device is used to generate a 5.2 V voltage rail, which most of the remaining power tree uses as input. The second output channel is used to generate the FPGA core voltage of 1.0 V, due to the high current that may be drawn by the FPGA on this rail.

The pre-regulator output voltage, among many other operating parameters, is configurable via PMBus. This feature is used to reduce the system power consumption and increase the system efficiency by allowing the FPGA to run at a slightly lower core voltage when supported. It can also be useful to increase the voltage headroom for some low-noise linear regulators that may offer better noise performance when operated in that way, if needed. The PMBus interface is available to the PMC and also to the FPGA, such that both devices can control the regulator, although only one may be the bus master at a given time.

The MIC23155 and MIC23159 general purpose step-down switching regulators are used to generate several voltage rails for the various system blocks. The MIC23155 is the single regulator IC version of the MIC23159. A 1.2 V rail, capable of sourcing up to 2 A, is generated to feed the FPGA high-speed serial transceivers termination voltage input. A 1.8 V rail is used to power the FPGA auxiliary logic, as required by the manufacturer. A 1.35 V is generated to power the DDR3 memory and the two associated interface pin banks on the FPGA, which are dedicated to communications with this memory. An MIC5166 memory terminator power supply integrated circuit is used to generate the termination voltage for the DDR3 address, command and control interface lines, and also takes care of the reference voltage generation for the input buffers. It requires a connection to the 1.35 V rail, but draws very little power from this rail, as it's only used as an internal reference. The bulk power is drawn from the 5.2 V rail.

One of the remaining FPGA pin banks is powered by a 3.3 V rail, and is mainly used to interface with the majority of the system peripherals, such as the mmWave

synthesizer, the audio CODEC, the power management unit and the clock manager. The last FPGA pin bank is powered by a 2.5 V rail, and is mainly used to interface with the RF transceiver IC, whose interface supply pin is also fed from the 2.5 V rail.

When designing a hybrid power distribution system, using a low-noise linear regulator after a switching regulator, it is crucial to consider its power supply rejection ratio (PSRR) performance across frequency to ensure effective noise filtering. The PSRR can be given by

$$PSRR = 20 \log \left(\frac{\Delta V_{OUT}}{\Delta V_{IN}} \right), \tag{3.5}$$

where ΔV_{OUT} is the variation in the output voltage of a device produced by the variation in the input voltage that originated it, ΔV_{IN}. As the switching regulator introduces ripple and noise at its operating frequency, harmonics, and, sometimes, sub-harmonics, the LDO's ability to reject this noise depends on its PSRR across the relevant frequency range. A high PSRR at the most critical and noisy frequency ranges ensures that most of the ripple is attenuated, delivering a clean output voltage. However, as PSRR typically decreases at higher frequencies, careful selection of the LDO and proper layout techniques, such as input and output filtering using low equivalent series resistance (ESR) capacitors, are essential to maximize noise suppression and achieve optimal performance in the sensitive power domains of the SDR.

As can be seen in fig. 3.8, the 2.5 V rail produced by the MIC23159 switching converter is powering two ADM7171 low-noise LDOs. These regulate the voltage down to 1.3 V, which then power the RF transceiver IC core. This design allows a very clean voltage rail to power the sensitive RF circuitry, while minimizing power loss by reducing the voltage difference between the ADM7171 output and input, as mentioned before. The 2.5 V rail is the lowest voltage rail in this design that can be used to power the ADM7171, since its minimum working input voltage is 2.3 V. Figure 3.10 depicts the typical PSRR performance of the ADM7171 across frequency. The previously mentioned behavior of a worse PSRR at higher frequencies can clearly be observed, but a good PSRR of about -45 dB is also visible at around 3 MHz. This is the switching frequency of the MIC23159 regulator that precedes the ADM7171 LDO. Higher order harmonics of the switching frequency will still be attenuated by the LDO, as the PSRR is about -30 dB at up to 10 MHz, but this higher frequency noise rejection is improved by the output filtering capacitors.

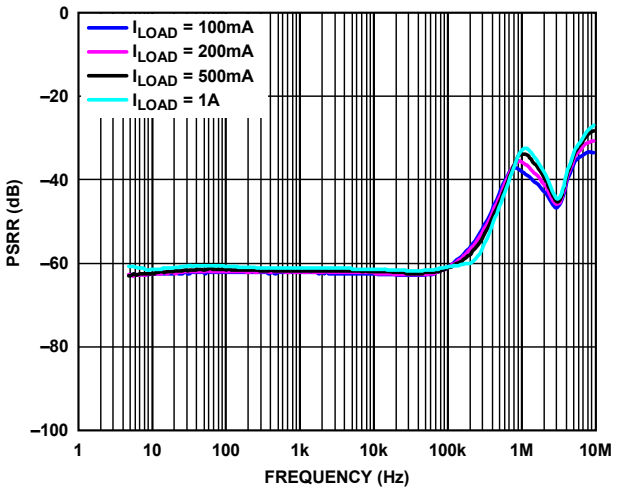


Figure 3.10: ADM7171 PSRR vs frequency, at different loads, with a headroom of 800 mV [52].

To power the mmWave synthesizer, again, two ADM7171 regulators are used. This time, as the IC requires 3.3 V, the regulators are fed by the 5.2 V rail. As the voltage headroom of 1.9 V is slightly higher than desired, it is decided to split the mmWave synthesizer load between two LDOs, in order to facilitate the transfer of the dissipated heat to the PCB.

The controllable integrated gain blocks on the RF transceiver outputs and the Bias-Ts are powered by a clean 5.0 V rail, generated by an ADM7172, powered by the 5.2 V rail. This is possible due to the very low dropout voltage of the ADM7172. The ADM7170, ADM7171 and ADM7172 belong to the same family of low-noise linear regulators, and are pin-compatible with each other. The main difference is the output current capability of 500 mA, 1 A and 2 A, respectively.

An ADP150 is used to power the audio CODEC with a clean 3.3 V source. It is used instead of an additional ADM7171 because the power consumption of the CODEC is lower. The ADP150 is capable of delivering up to 150 mA of current.

3.2.6 *Expansion card interconnections*

The interface between the base SDR platform and the expansion cards is of paramount importance and cannot be overlooked. It is designed to offer as much flexibility as possible (e.g., power, communication, clock), without compromising the functionality of the base platform.

The interface consists of three groups of 2.54 mm spaced contact points, each of the groups being organized in a 2 by 3 grid, as depicted in fig. 3.11. One of the contact groups is entirely dedicated to delivering power to the expansion card, where three voltage rails are available, as illustrated in fig. 3.11a. Two contacts provide 5.2 V, directly from the pre-regulator of the PMU, while a third and fourth contacts provide 2.5 V and 3.3 V, respectively, from switching regulators. The last two contacts in this group carry the return current, and are connected to the reference potential, i.e., ground. Each contact is rated to carry up to 3 A, which allows an expansion card to consume up to 6 A of current, limited by the two ground contacts. This is, of course, limited by the regulator's current output capability, so the designed current limit of the contacts should never be hit.

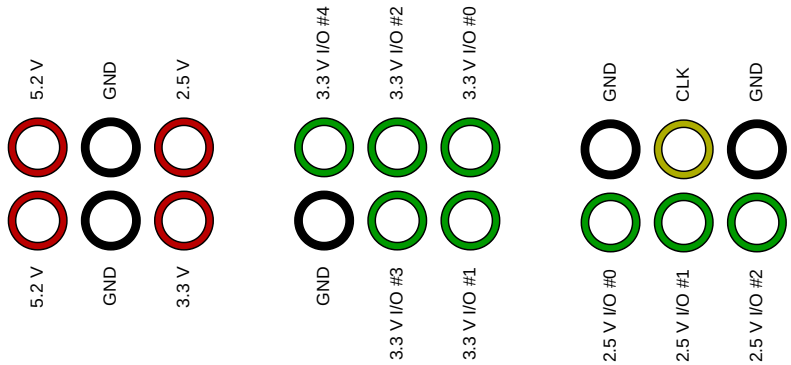


Figure 3.11: Expansion card connectors pin diagrams: (a) Power connections; (b) 3.3 V GPIO connections; (c) 2.5 V GPIO and clock connections.

Regarding communications with the expansion card, there is a single group of connections, fig. 3.11b, dedicated to five 3.3 V LVCMOS GPIOs directly connected to the FPGA, along with a ground connection for the return current. This connector is not intended for high-speed logic signals, but to carry slow management interfaces such as I2C or infrequently switched control lines. Another set of three 2.5 V LVCMOS GPIOs are available on the last group of contacts, as represented in fig. 3.11c. The remaining three lines of this last group is arranged in a ground-signal-ground (GSG) manner, and provide a direct clock line connected to the clock management unit.

A GSG connection layout offers several advantages, particularly in high-frequency applications, as expected for the clock signal. By placing ground pads on either side of a signal pad, GSG layouts help control signal integrity by reducing parasitic effects and providing a well-defined return path for high-speed signals. This minimizes crosstalk, lowers signal distortion, and improves impedance matching. Additionally,

the symmetric ground placement enhances EMI suppression and reduces the risk of unwanted coupling between adjacent lines [53].

3.3 PCB STACKUP AND LAYOUT

Selecting the PCB stackup for the SDR base platform required balancing RF performance, impedance control, and cost. While some RF traces are present, the majority of them consists of short and constant impedance lines connecting pre-matched components to each other and/or to connectors, meaning that complex impedance matching networks are not necessary on the PCB itself. Ideally, a PTFE/Teflon-based high-frequency dielectric would provide superior signal integrity, lower dielectric loss, and minimal signal dispersion at high frequencies. However, such materials significantly increase manufacturing costs and may impose tighter fabrication constraints, making them less practical for this application. Instead, a compromise was made using an FR-4 core and prepreg materials that ensure controlled impedance while keeping fabrication costs reasonable. By carefully selecting materials with well-characterized dielectric properties, it was possible to maintain good signal integrity while ensuring manufacturability using standard PCB fabrication processes.

Typically, an 8-layer PCB would be preferred for an extensive design as is the base SDR platform, to achieve optimal isolation, power integrity, and routing flexibility. Additional layers would allow improved shielding between signal layers, reducing crosstalk and minimizing unwanted coupling. However, to keep fabrication costs within a reasonable budget, a 6-layer stackup was chosen instead. Despite this reduction in layer count, careful layer arrangement and routing strategies ensure that the 6-layer configuration remains easily manageable, providing sufficient isolation for RF traces while maintaining controlled impedance for high-speed signals. The final stackup is detailed in table 3.1 and represents a practical balance between performance, manufacturability, and cost-effectiveness, ensuring that the PCB meets both electrical and mechanical requirements without unnecessary complexity.

During the initial PCB layout phase, components were strategically placed to ensure short, direct connections between elements that needed to interface closely. Key components were positioned with functional grouping in mind, minimizing trace lengths to reduce parasitic effects and signal degradation. RF components, connectors, and high-speed interfaces were arranged to optimize signal flow while avoiding unnecessary routing complexity, as can be seen in fig. 3.12, which depicts

Table 3.1: Base SDR platform PCB stackup.

Layer	Material Type	Thickness
Top	Copper	35 μm
	Prepreg 3313	0.0994 mm
Layer 2	Copper	15.2 μm
	FR-4 Core	0.55 mm
Layer 3	Copper	15.2 μm
	Prepreg 2116	0.1088 mm
Layer 4	Copper	15.2 μm
	FR-4 Core	0.55 mm
Layer 5	Copper	15.2 μm
	Prepreg 3313	0.0994 mm
Bottom	Copper	35 μm

a practically fully assembled PCB prototype with the functional component groups identified. By prioritizing proximity and logical placement, it is possible to achieve a cleaner layout, reducing trace meandering, ultimately enhancing signal integrity and manufacturability. After roughly arranging the key components of the circuit functional blocks, the trace routing task can be started.

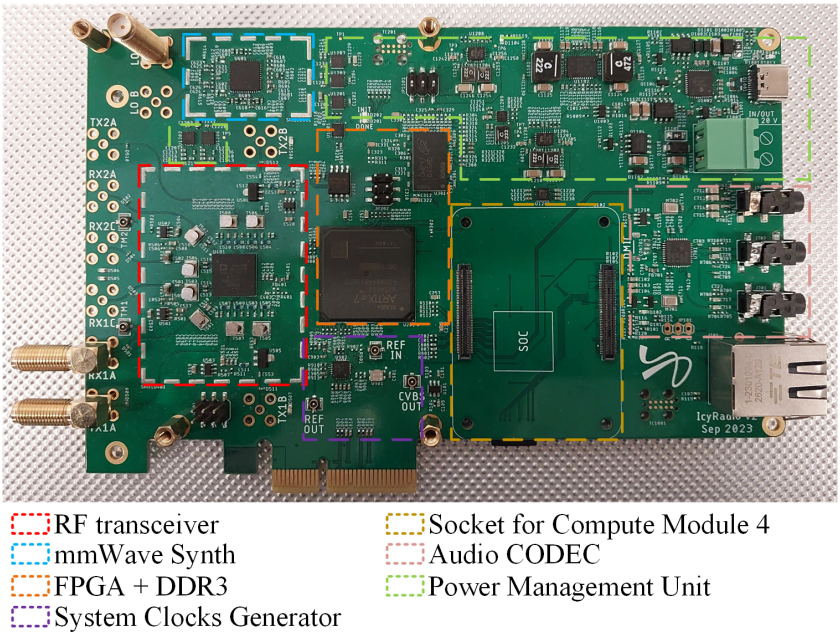


Figure 3.12: Assembled PCB prototype of the base SDR platform with identified functional sections.

One of the routing challenges is the DDR3 memory interface, which connects the FPGA to the memory IC, which requires careful placement of traces, and cannot be overlooked. Length matching when routing a DDR3 interface is crucial to ensure signal integrity, minimize timing skew, and achieve reliable high-speed data transfers. DDR3 memory operates with strict timing requirements, and mismatches in trace lengths can lead to data corruption, reduced performance, or complete communication failure. The most critical signals in a DDR3 interface include the clock, address/command/control, and data (DQ) buses, including the data strobe (DQS), each of which requires careful routing to meet the necessary timing constraints [54]. A summary of the trace length constrains which governed the DDR3 interface routing can be observed on fig. 3.13.

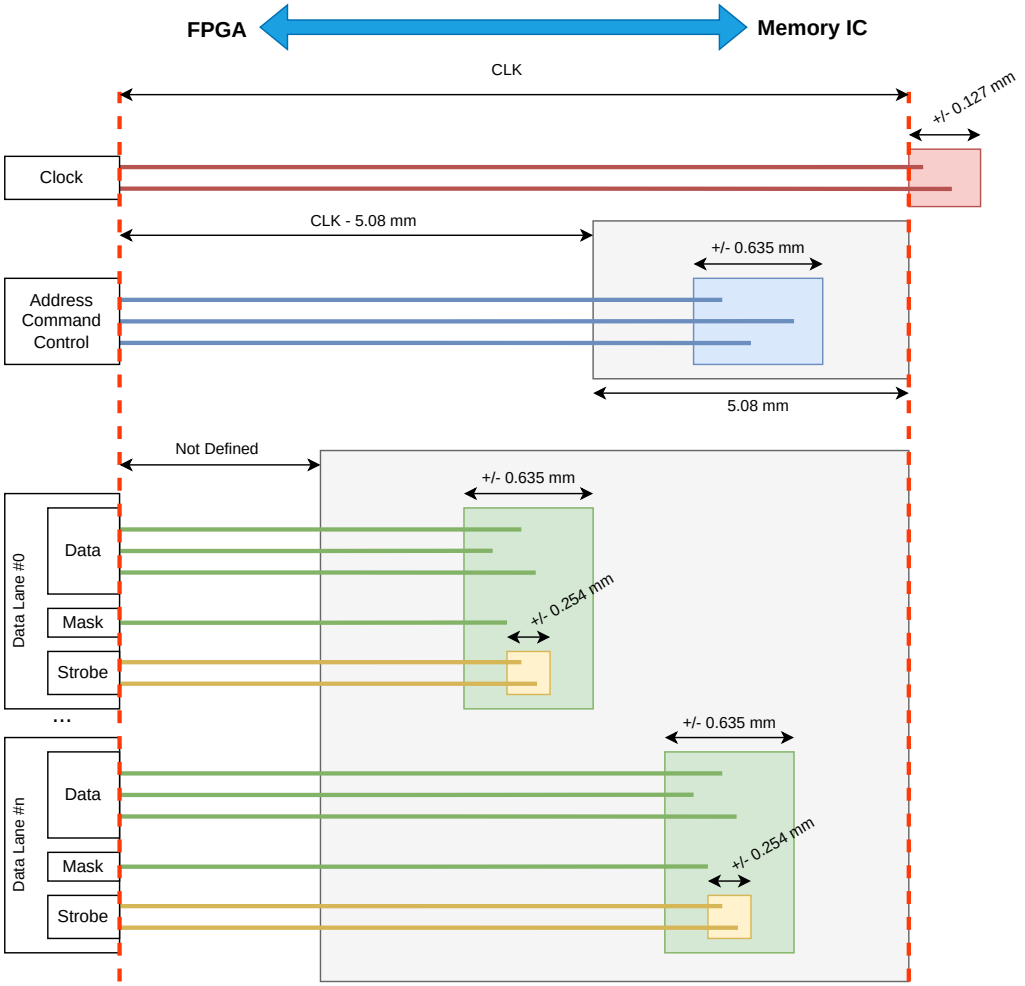


Figure 3.13: Summary of trace length matching rules for the DDR3 interface.

The primary reason for length matching is to ensure that signals arrive at the memory device or controller at the correct time relative to each other. Since signals travel at different speeds depending on the PCB material and layer stack-up,

any variation in trace length can cause timing misalignment. This is particularly important for the data bus, where signals must be aligned within a tight timing window, known as the setup and hold time, to be sampled correctly by the receiving device. If signals arrive too early or too late, the data may be misinterpreted, leading to read and write errors [55].

The DDR3 clock signal must be carefully matched with the address, command and control signals to ensure proper synchronization. If the clock arrives too early or too late relative to these signals, the memory may latch incorrect address or command data. Similarly, within a byte lane, the DQ signals must be matched with the corresponding DQS signal to maintain correct data timing. DDR3 uses a source-synchronous clocking scheme, where the DQS signal aligns with data during transmission. If length mismatches exist, some bits may arrive at the receiver too soon or too late, causing bit errors [55].

A valuable reference for selecting the routing topology and defining the constraints is the Xilinx provided application note [56]. Following the guidelines provided on that document, among with the vast knowledge acquired from [54], the memory interface was routed using stripline transmission lines in the internal layers 3 and 5 of the PCB. Striplines are formed by the traces at the aforementioned layers, using reference planes at layers 2, 4 and, in the interface area, also on the bottom layer (layer 6). A snapshot of the routed DDR3 interface is shown in fig. 3.14, where the meanders that provide length matching can clearly be seen.

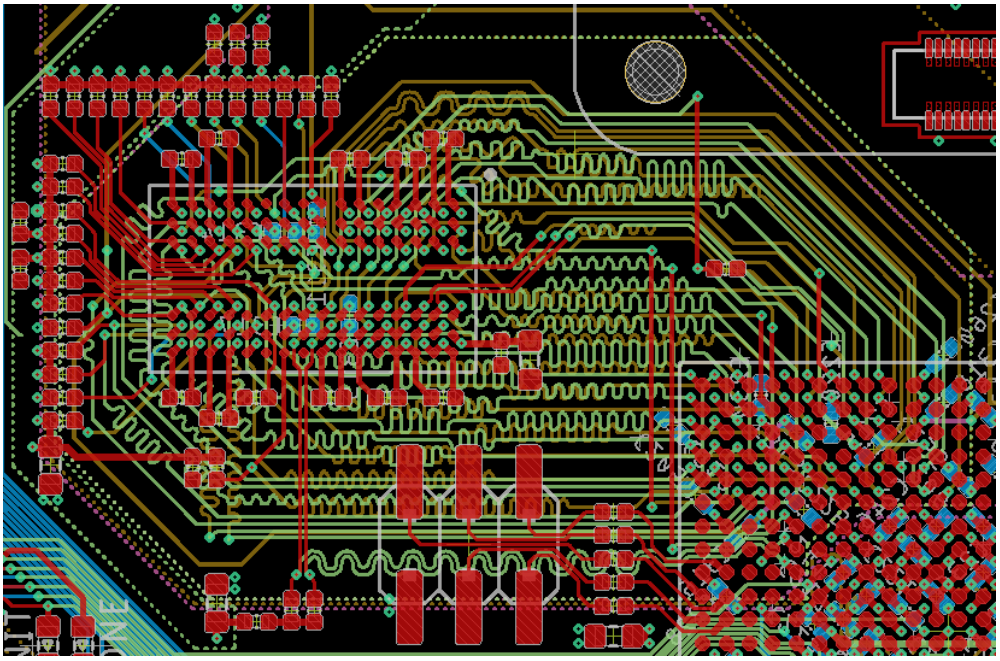


Figure 3.14: Snapshot of the DDR3 trace layout and length matching.

Also of increased importance for the correct performance of the SDR is the RF transceiver parallel data interface. This interface comprises two 12-bit ports, two clock signals and two framing signals, as depicted in fig. 3.2. Length matching of these traces is not so critical, as the transceiver IC implements configurable delay elements which allow timing mismatches to be corrected. The parallel interface between the AD9361 and the FPGA is depicted in fig. 3.15, where the source-synchronous nature of both 12-bit data ports can be seen. On the received data

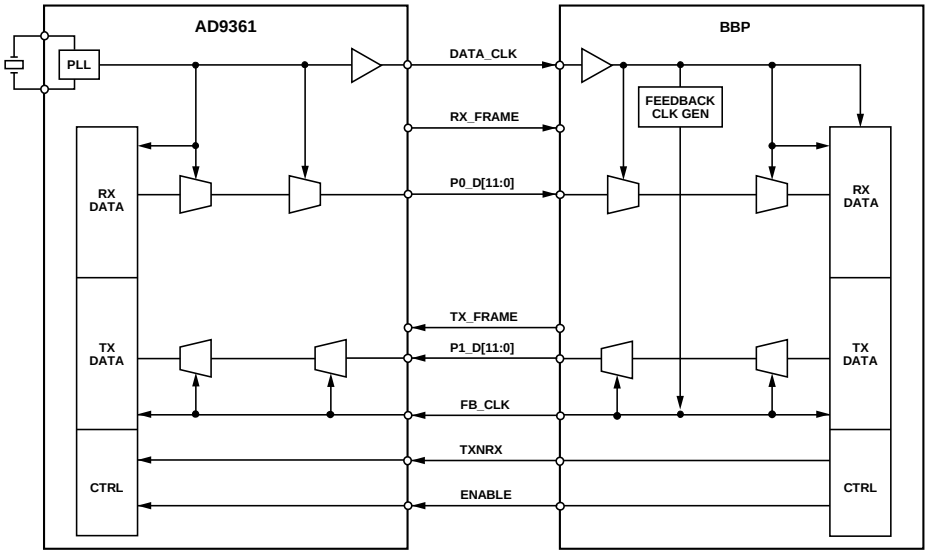


Figure 3.15: Interface connections between the RF transceiver and the FPGA [57].

port, the RF transceiver actively drives the data and clock ($DATA\_CLK$), which the FPGA receives. On the transmitted data port, the FPGA derives a clock from the clock signal received via the RX data port, and actively drives the 12 bits of data synchronous to that derived clock (FB_CLK). The fully routed parallel data ports, along with clocks, framing and other general purpose and slower control lines can be observed in fig. 3.16. Layer 3 of the PCB is used to route these signals as stripline transmission lines.

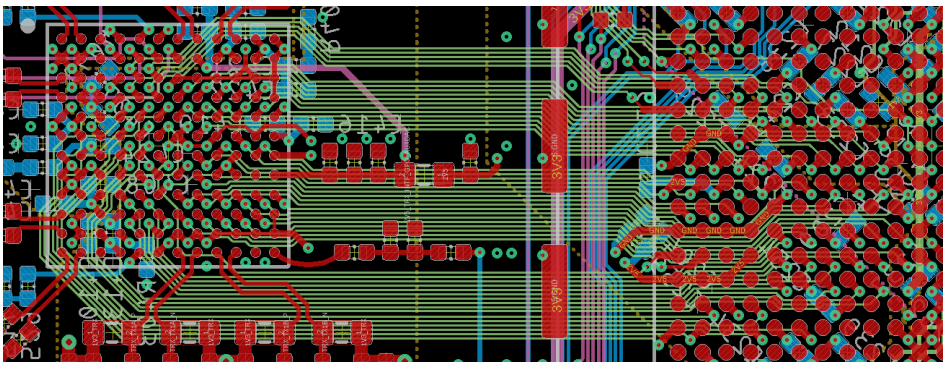


Figure 3.16: Snapshot of the RF transceiver parallel data interface routing.

The power management unit section of the SDR, mostly condensed on the top right corner of the PCB, as seen in fig. 3.12, is further subdivided in small circuits which comprise each regulator depicted in fig. 3.8 along with all the passive components required for their operation, which range from a couple of filtering capacitors for linear LDOs to a slightly more complex layout for the switching regulators.

Power integrity (PI) is a critical aspect of the PCB layout, ensuring that power delivery networks (PDNs) provide stable and noise-free voltage to the relevant circuits. Poor power integrity can lead to signal degradation, increased electromagnetic interference, and system malfunctions [58]. Maintaining PI involves minimizing voltage fluctuations and transients, reducing power supply noise, and ensuring PDNs present a low impedance across different frequencies [59].

In switching converters, minimizing the power switching loop area is crucial for enhancing efficiency and reducing EMI. A smaller loop area diminishes parasitic inductance, leading to reduced voltage spikes and switching losses. This practice not only improves the converter's performance but also mitigates EMI, ensuring compliance with regulatory standards and minimizing potential interference with nearby electronic devices [60]. An example of a typical component placement which minimizes the high-frequency switching current loop is depicted in fig. 3.17, which shows the layout of both channels of the LT7182S pre-regulator, with the rough current loop identified for one of the channels.

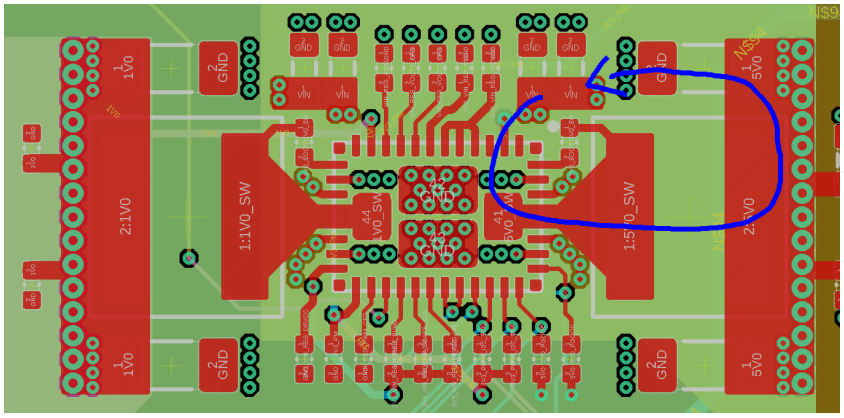


Figure 3.17: Snapshot of the pre-regulator routing, with the high-frequency signal loop identified.

Bulk power transmission through the PCB is achieved through copper pours, mostly in the internal PCB layers, which span the circuit areas that connect to a particular power rail. The return current path is guaranteed by the PCB layer 2, which is completely filled with a ground pour, and also by several other layers

where free space has been also filled with ground pours, properly stitched to layer 2 with vias. The global network of power copper pours can be seen in fig. 3.18.

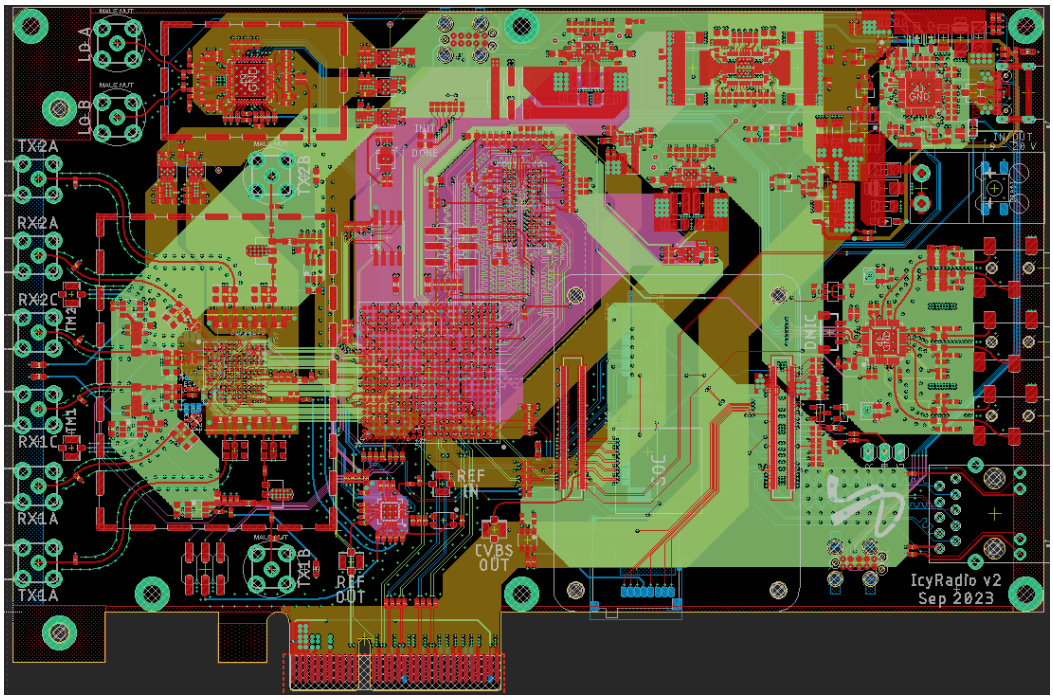


Figure 3.18: Snapshot of the base SDR platform PCB with all the power pours filled.

4

SDR BASE PLATFORM FIRMWARE AND SOFTWARE DEVELOPMENT

Software defined radio technology has revolutionized the field of wireless communications, offering unprecedented flexibility and adaptability in radio systems. By implementing radio functions in software rather than hardware, SDRs enable rapid prototyping, multi-standard operation, and dynamic reconfiguration of radio parameters. This chapter delves into the intricate details of firmware and software development for the SDR base platform, exploring the complex interplay between hardware, firmware, and software components that form the backbone of the SDR base platform. The chapter examines the digital architecture implemented in the FPGA, addressing the design of key logic blocks, including custom and proprietary IP cores specifically developed for this project, and covers the software stack, including the Linux device drivers and their interface with userspace, which together enable the creation of a versatile and powerful SDR. The chapter spans everything from low-level FPGA firmware implementations to high-level user applications, providing a comprehensive overview of the technical challenges and solutions involved in developing a state-of-the-art SDR system.

4.1 FPGA DESIGN ARCHITECTURE

The FPGA of the SDR base platform serves as a data processing unit, system management and central hub for various interfaces, promoting communication, data transfer, control and monitoring between different functional blocks, as illustrated in fig. 3.1, ensuring seamless operation of the SDR platform. As can be seen, the FPGA connects to the AD9361 RF transceiver, which handles digital filtering, signal domain conversion, frequency conversion, and signal amplification. The FPGA communicates with the Raspberry Pi CM4 via PCIe, enabling high-speed data exchange and allowing digital signal processing to occur at the software level. Additionally, the FPGA interfaces with DDR3 memory to enhance the system's flexibility and performance. Other interfaces include I2C and SPI for management, control and data transfer, I2S for audio data transfer, and Ethernet for network

connectivity. The FPGA also controls GPIO lines and interfaces with the power management unit.

In modern FPGA-based development, The Vivado IP Integrator provides a powerful visual framework for assembling complex systems using pre-designed IP cores, streamlining hardware design while maintaining flexibility for software-driven signal processing. This section explores how the IP Integrator was leveraged to create the architecture that is, effectively, the brain of the SDR, by integrating multiple IP cores, from different vendors, including blocks specially designed for this purpose, as presented in the previous sections.

To accomplish all the above-mentioned functions, an FPGA logic design capable of data processing and digital bus communication for management, control and data transfer must be developed and implemented, as illustrated in fig. 4.1. This high-level bus block diagram outlines the architecture and components (and their interconnections) involved in the firmware (and supporting software) development for the FPGA used in the SDR base platform. Key elements include interfaces and controllers such as GPIO, (Q)SPI, I2C, I2S, and PCIe for high-speed communication and data transfer. Memory and storage components include a DDR3 interface, a BRAM scratchpad, and a memory-mapped SPI flash for non-volatile storage. The processing units feature a PicoRV32 RISC-V core and multiple DMA controllers to handle data transfers with minimal CPU intervention. Communication and data management are handled by interconnected AXI buses, AXI stream point-to-point interfaces, an interface IP for the RF transceiver. An RF timestamping core is included to handle precise radio timing. System management components include reset and clock management logic, an interrupt controller, and a non-volatile device information core. Additional components include interconnections with a mmWave synthesizer and an audio CODEC. The FPGA design also considers the interface with expansion cards for added functionality. This comprehensive overview highlights the integration of various elements to support SDR applications.

In modern FPGA-based development, the Vivado IP Integrator provides a powerful visual framework for assembling complex systems using pre-designed IP cores, streamlining hardware design while maintaining flexibility for software-driven signal processing. This section explores how the IP Integrator was leveraged to create the FPGA logic architecture that is, effectively, the brain of the SDR, by integrating multiple IP cores, from different vendors, including blocks specially designed for this radio.

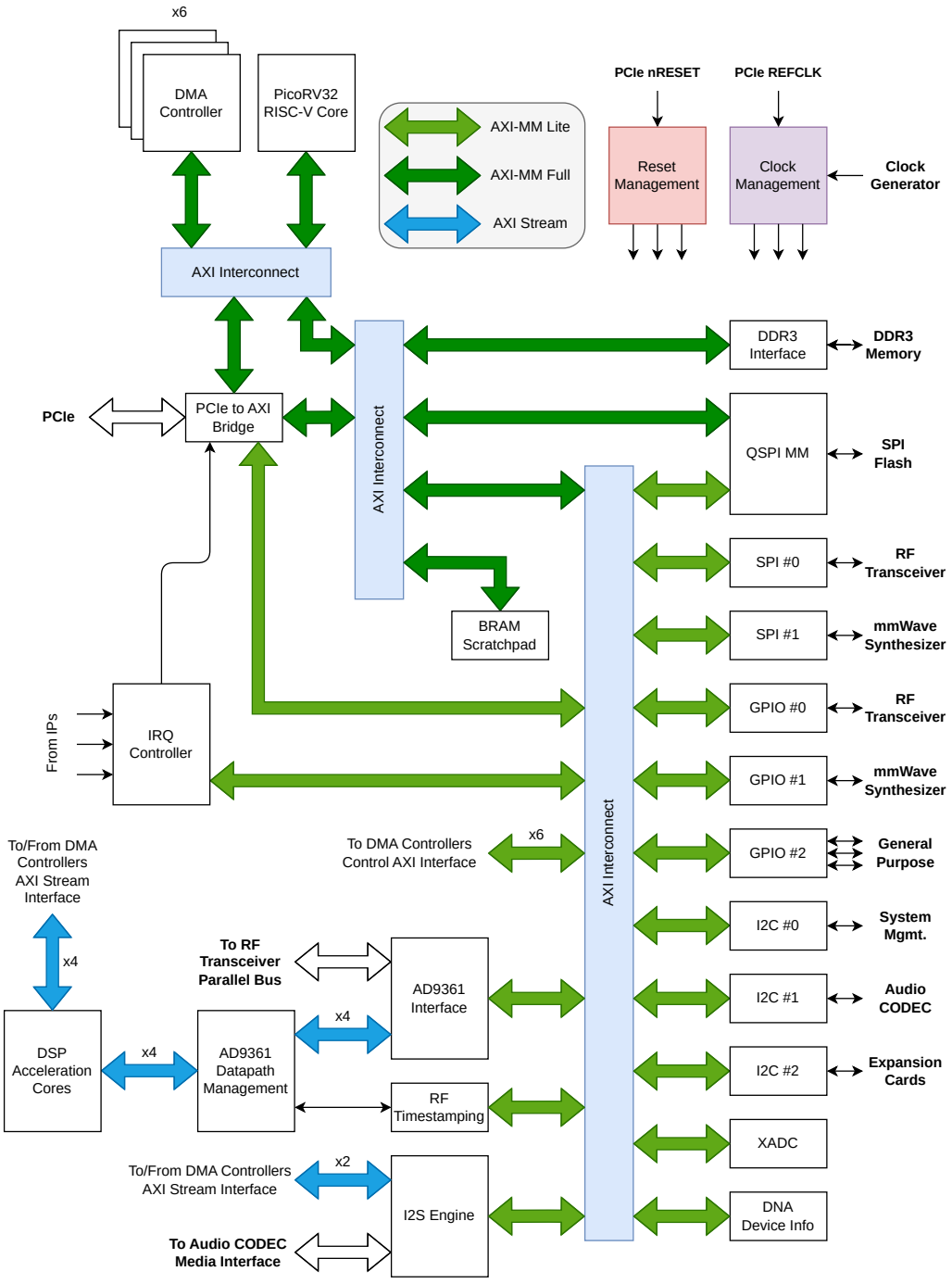


Figure 4.1: FPGA logic design high-level block diagram.

The logic design employs many modern interfaces and techniques, as described above and presented in fig. 4.1. The key IP cores that comprise the FPGA logic design are listed in table 4.1. Their usage, interconnections between IPs, as well as the challenges faced during the development are further addressed throughout this section. As can be seen, there are several IPs from Xilinx, one from Wolf, C [61], several from Analog Devices and nine proprietary IPs, developed specifically for integration in this platform, to improve the functionality and flexibility of the SDR.

Table 4.1: Principal IP cores comprising the FPGA logic design, with resource usage and brief purpose description.

Core	**Vendor**	**LUT**	**FF**	**BRAM**	**Purpose**
PicoRV32	Wolf, C. [61]	2205	996	0	General purpose system management.
XADC Wizard	Xilinx	263	371	0	XADC hard IP interface.
MIG	Xilinx	4804	4195	0	DDR3 interface.
Clocking Wizard	Xilinx	1	0	0	Clock synthesis for DDR3, MMCM hard IP interface.
AXI MM to PCIe	Xilinx	12582	10245	16	PCIe interface with PC or CM4.
AXI I2C 0	Xilinx	415	385	0	Audio CODEC control.
AXI I2C 1	Xilinx	415	385	0	General purpose system control.
AXI I2C 2	Xilinx	415	385	0	Expansion cards control.
AXI DMA 0	Analog Devices	437	730	2	RF Transceiver TX channel 0 data streaming.
AXI DMA 1	Analog Devices	432	731	2	RF Transceiver TX channel 1 data streaming.
AXI DMA 2	Analog Devices	370	728	1	RF Transceiver RX channel 0 data streaming.
AXI DMA 3	Analog Devices	371	728	1	RF Transceiver RX channel 1 data streaming.
AXI DMA 4	Analog Devices	428	731	2	I2S TX data streaming.
AXI DMA 5	Analog Devices	469	817	1	I2S RX data streaming.

Core	Vendor	LUT	FF	BRAM	Purpose
AXI AD9361	Analog Devices	9511	12673	0	RF Transceiver parallel data bus interface.
AD9361 DAC Unpacker 0	Analog Devices	27	44	0	RF Transceiver TX channel 0 datapath formatting.
AD9361 DAC Unpacker 1	Analog Devices	27	44	0	RF Transceiver TX channel 1 datapath formatting.
AD9361 ADC Packer 0	Analog Devices	29	47	0	RF Transceiver RX channel 0 datapath formatting.
AD9361 ADC Packer 1	Analog Devices	29	47	0	RF Transceiver RX channel 1 datapath formatting.
AXI SPI 0	**This work**	162	117	0	RF Transceiver control.
AXI SPI 1	**This work**	162	117	0	mmWave Synthesizer control.
AXI QSPI MM	**This work**	483	410	0	SPI Flash interface.
AXI RF Timestamping	**This work**	517	2494	0	RF Transceiver sample timing control.
AXI IRQ Controller	**This work**	1453	474	0	IRQ multiplexing and routing.
AXI GPIO 0	**This work**	110	195	0	RF Transceiver I/O control.
AXI GPIO 1	**This work**	110	195	0	mmWave Synthesizer I/O control.
AXI GPIO 2	**This work**	110	195	0	General purpose system I/O control.
AXI I2S	**This work**	207	403	0	Audio CODEC media interface.

Figure 4.1 shows that the main interconnection mechanism utilized in the FPGA design are AXI interfaces. The use of AXI in the design, particularly those that employ a vast portfolio of IP cores, is justified by several technical and practical advantages. AXI provides a standardized, high-performance interface that ensures efficient data transfer, modular design, and seamless integration of custom and third-party IP cores.

One of the primary reasons for using AXI interfaces is their scalability and modularity. The AXI memory-mapped interface enables easy integration of multiple peripherals in a hierarchical design, allowing efficient interconnection between processor cores (e.g., ARM Cortex-A in Zynq SoCs, or the PicoRV32 [61] soft-core) and custom accelerators. Similarly, AXI Stream interfaces facilitate high-speed streaming data transfers between processing elements, such as DSP blocks, without the overhead of address-based transactions. These modular capabilities significantly reduce design complexity.

AXI's well-defined protocol simplifies the integration of new hardware components, reducing design time and verification effort in FPGA-based systems. The ability to instantiate and connect multiple IP cores without requiring major modifications to the overall architecture makes AXI the preferred choice in modern FPGA-based designs as is the case with this software defined radio [62]. Furthermore, Xilinx's Vivado IP integrator provides built-in AXI interconnects that simplify the integration of custom and third-party IP core, ensuring that different functional blocks can be interconnected efficiently. Additionally, the block design environment allows visually connecting AXI-based IP cores without manually handling low-level signal wiring, which highly facilitates development of complex logic designs. Figure 4.2 provides a comprehensive visualization of the address map of the global AXI memory-mapped network, which globally interconnects all the IPs which expose AXI memory-mapped master and slave interfaces. Figure 4.2a provides a global overview of the entire address space, while fig. 4.2b details the peripheral IP configuration register region.

Table 4.2 summarizes the logic resource usage of the entire FPGA design, and it is possible to observe that more than 35% of the LUTs (the resource with highest utilization) are available for design expansion, such as additional DSP algorithm implementations, aligning with the expandability objective of the SDR platform.

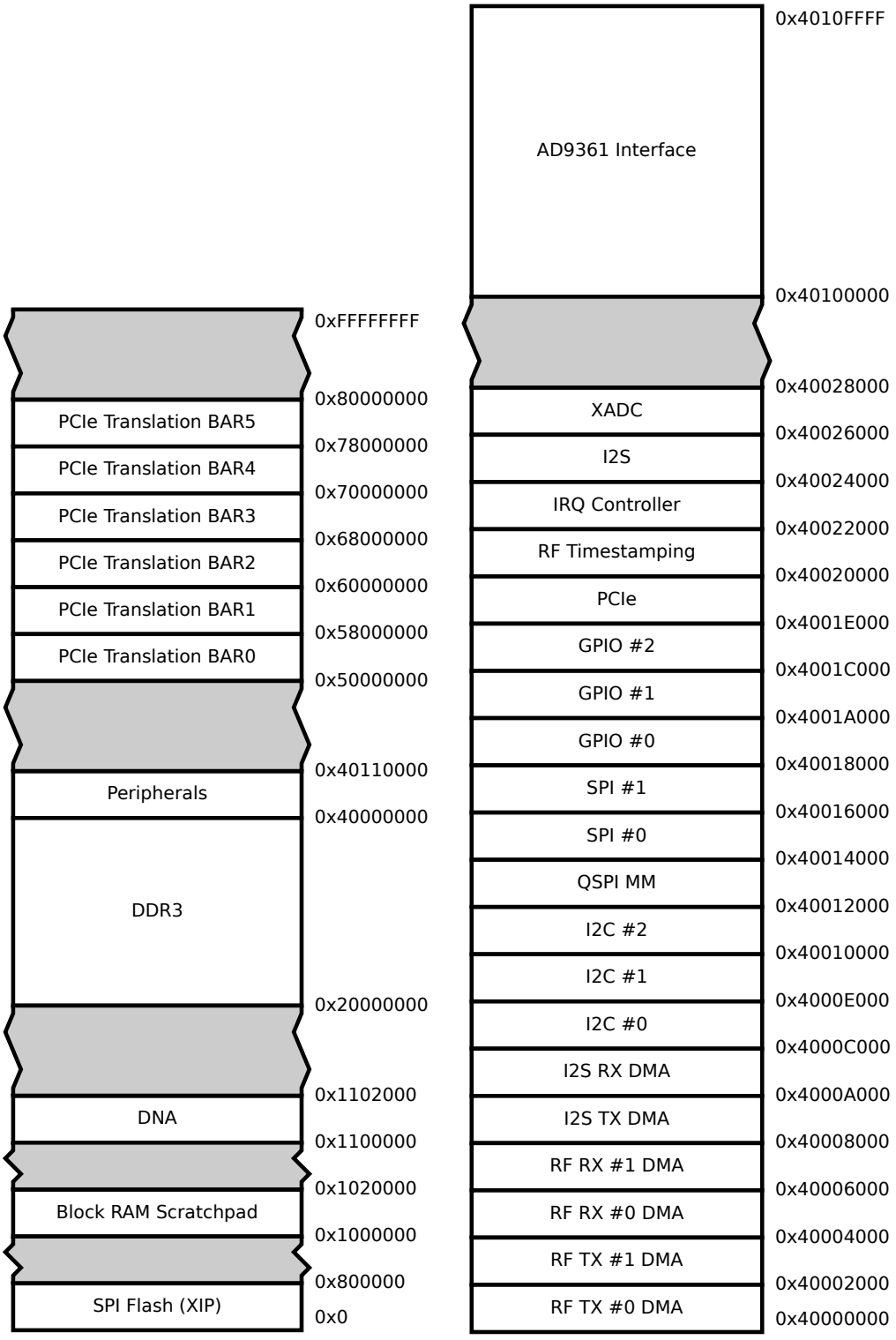


Figure 4.2: AXI bus address map: (a) Main address space; (b) Detailed peripheral region.

Table 4.2: FPGA design resource usage on a Xilinx Artix-7 A100T.

Resource	**Utilization**	**Available**	**Utilization (%)**
LUT	40597	63400	64.03
LUTRAM[1]	1936	19000	10.19
FF	46374	126800	36.57
BRAM	71.5	135	52.96
DSP	32	240	13.34
IO	138	285	48.42
GT	2	4	50.00
BUFG	10	32	31.25
MMCM	3	6	50.00
PLL	1	6	16.67
PCIe[2]	1	1	100.0

[1] LUTs used as RAM.
[2] Hard (i.e., fixed function) IP block on the package.

4.2 CUSTOM IP CORES

Despite the large repository of IP cores offered by Xilinx, not all the functionality required in this project can be performed by such cores. In addition, some cores that, on the surface may seem suited for a specific task, in reality show some deficiencies and/or lack required features.

To overcome the aforementioned scenarios, several IP cores were developed, which will be discussed next. All the cores are written in Verilog hardware describing language (HDL) and testbenches have also been provisioned to perform basic evaluation of their functionality.

4.2.1 *AXI I2S Engine*

An IP core was developed to interface with the utilized audio CODEC, which primarily communicates with an audio processor through an I2S interface. Its functional block diagram is given in fig. 4.3. The core is segmented as much as possible, such that each Verilog *always* block can be correlated with a specific task. As can be seen, the core has three main blocks, namely, the I2S serialization and deserialization, the clock divider and the AXI register page block. As illustrated in fig. 4.3, the core operates in two different clock domains, the AXI clock domain

and the I2S clock domain, thus, clock domain crossing (CDC) logic is necessary to minimize the risk of metastability when transferring signals between the two asynchronous clock domains. Since most of the logic is synchronous to the I2S clock, with the AXI clock domain being mostly used to drive the AXI4-Lite memory mapped interface logic, metastability in CDC signals is avoided by employing a multi flip-flop synchronizer in the signal's path. As all signals that cross clock domains are single-bit registered signals, no additional measures are required, i.e., to handle multi-bit signal vectors. The AXI I2S engine IP has three main interfaces, an AXI4 Lite interface for reading and writing a set of registers (detailed in appendix B.1) and two AXI4-Stream interfaces (a master and a slave) for carrying raw audio samples to and from the serial I2S bus, as illustrated in fig. 4.3.

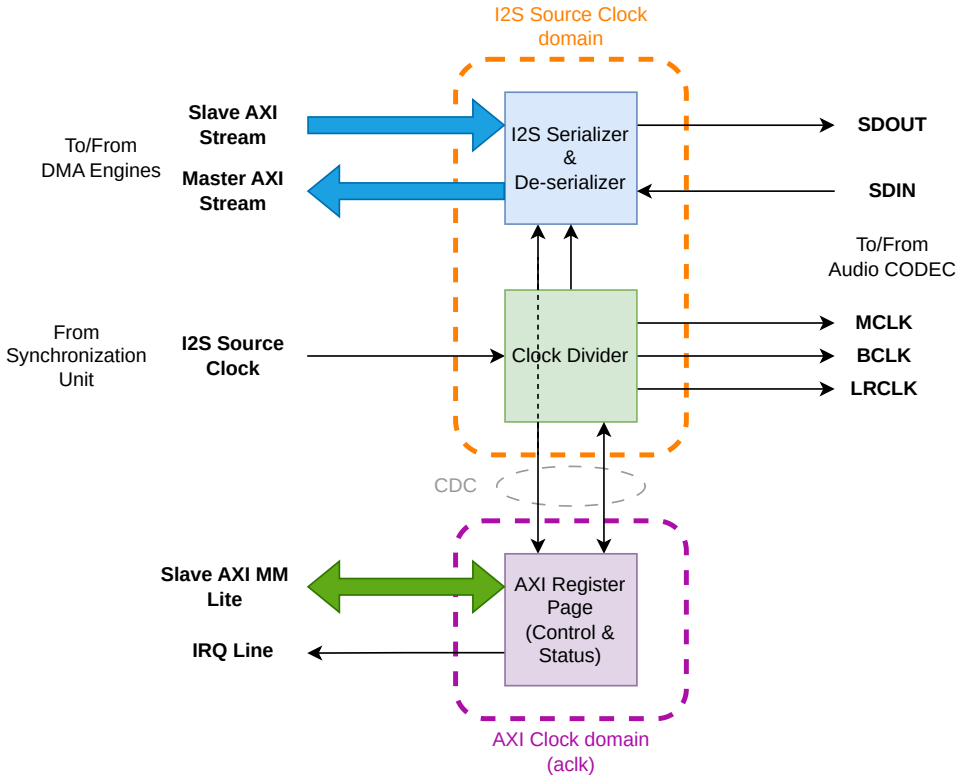


Figure 4.3: I2S engine IP core functional block diagram.

The clock divider block of fig. 4.3 comprises two similar clock dividers to generate the *BCLK* and *LRCLK* I2S interface clocks. An additional divider is used to generate a *MCLK* signal, which is not strictly needed by the I2S interface, but serves as a timing clock for the audio CODEC (ADCs, DACs and DSP logic). All dividers employ the same logic: a counter is incremented every rising edge of the I2S source clock, the counter value is constantly compared to a set value, the counter *TOP* value, and once this value is reached, it is reset back to zero. When the counter

is reset to zero, a signal is toggled. This signal is the output of the divider, i.e., *BCLK*, *LRCLK* or *MCLK*. The division factor is configurable at runtime to any even integer multiple of the divider source clock.

Two asynchronous multiplexers are employed to implement the loopback feature. This feature is included in the core for testing purposes, and is generally turned off during normal operation, as it completely isolates the serial data input and output pads and connects them internally.

The main I2S serialization and deserialization logic is governed by a finite state machine (FSM), whose simplified state diagram is depisted in fig. 4.4. An enable

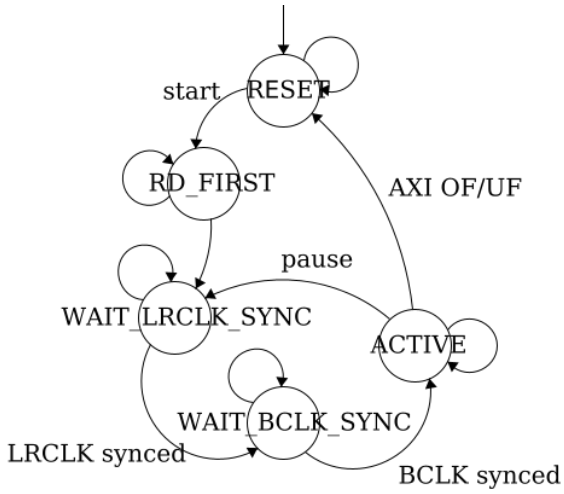


Figure 4.4: I2S serializer and deserializer FSM.

signal, controlled through the AXI4-Lite register interface, resets this state machine, and keeps it in a reset state until enabled. In this state, the core applies back-pressure in the slave AXI-Stream interface and does not output any data in the master AXI-Stream interface. Data on the I2S bus is also ignored, but the clock signals may or may not be toggling, as the clock dividers are gated by a separate enable signal. When first enabled, the FSM moves to the *RD_FIRST* state. In this state, it asserts *tready* on the slave AXI-Stream interface and waits for a data word to be received (i.e., both *tready* and *tvalid* asserted in the same clock cycle). After acquiring the first data word, containing either one 32-bit or two 16-bit audio samples, the machine needs to align itself with the *LRCLK* falling edge, which indicates the start of an I2S frame, as depicted in fig. 4.5. The *WAIT_LRCLK_SYNC* FSM state performs this synchronization. The *LRCLK* falling edge should coincide with the *BCLK* falling edge, otherwise the clocks are not properly aligned. Data from the I2S master is setup on the falling edge of *BCLK*, as shown by fig. 4.5, but not when this falling edge coincides with the *LRCLK* falling edge. So, a second synchronization step is

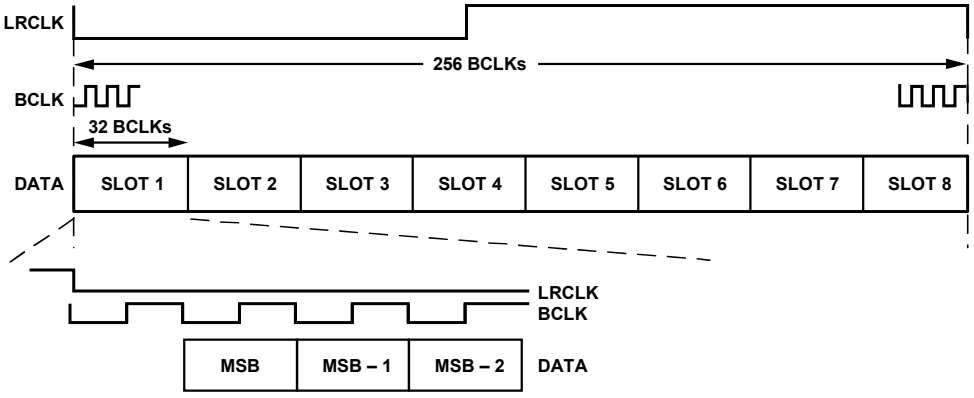


Figure 4.5: ADAU1372 I2S timing diagram [63].

required, *WAIT_BCLK_SYNC*, in which the machine waits for a rising *BCLK* edge.

After the two synchronization states, the machine is ready to start I2S transactions, so it moves to the *ACTIVE* state. In this state, there is only activity when *BCLK* changes state. When it falls, the output shift register is shifted, setting up a new bit in the data output line. When it rises, the input data bit is sampled, and shifted into its own shift register. When the output shift register is about to become empty, a new data word is requested from the slave AXI-Stream interface. Similarly, when the input shift register is full, the data word is output to the master AXI-Stream interface and *tvalid* asserted, indicating to a master that valid data is present on the bus and can be read. If, at any time, the AXI-Stream transactions are not completed in time, the whole machine is reset, and the process starts over. This is because the core has no buffering capability whatsoever, relying solely on the AXI-Stream endpoints to be fast enough at reading and providing data to this core. This is usually not a problem, since the core is designed to be interfaced with DMA engines in both AXI-Stream interfaces and such engines generally have buffers of their own. In addition to the described functionality, the core also supports being paused. The core behavior when commanded to pause is similar to when data overflows or underflows, with the exception that, in the pause state, the shift registers and AXI-Stream logic is not reset.

The core is able to generate interrupt requests to an eventual companion processor that may control it. It does so by setting different bits when certain pertinent actions happen, such as the loss or acquisition of sync between different clocks, or FSM state switching. These bits are then ORed together and a single request signal is generated. The receiver of this interrupt request can then read the individual

interrupt bits via the AXI4-Lite interface to figure out which event(s) triggered the interrupt.

All the aforementioned control and status signals to the different logic portions of the core are managed by the AXI master. The master accesses those bits via 32-bit registers, and the core has a dedicated logic section which implements the AXI memory-mapped handshake protocol, required to complete read and write transactions from and to those registers.

4.2.2 *AXI SPI Engine*

Although Xilinx provides a free-to-use and configurable quad SPI IP core, it is very limited in functionality when it comes to the runtime programmability. Most configuration options are only available as synthesis options, meaning they cannot be changed on-the-fly when the design is running. Such inconvenience motivated the development of an SPI IP core, which could fulfill all the requirements for this design, namely: the ability to operate in single (SPI), dual and quad IO modes, and the possibility to switch between modes in runtime; offering memory mapped accesses to a flash memory (execute in-place (XIP) mode) while being able to enter and exit such access modes in runtime, so that regular commands could be sent to the flash memory, an impossible operation when using Xilinx's IP core; and being able to configure the SPI clock frequency in runtime. Given this set of goals, and the two unique use cases in this project (interfacing with SPI flash memory and with regular SPI peripherals), three versions of the core were created: An *axi_qspi_mm* core, which is the fully featured core, with all the functions mentioned above; an *axi_qspi* core, similar to the first, but lacking the memory mapped access mode; and an *axi_spi* core, similar to the previous one, but lacking dual and quad IO modes.

All the cores support the four SPI modes, i.e., the four combinations of *CPOL* ans *CPHA*, as shown by fig. 4.6. DDR mode is not supported. The functional block diagram of the implemented SPI engine IP is shown in fig. 4.7.

The multi-version approach allows saving some logic resources, when not all the features of the core are required (i.e., no need to access a simple SPI peripheral using memory mapped interfaces). Granted, cores with more features can still be programmed to behave as the versions with less features during runtime. For completeness, the *axi_qspi_mm* core will be covered here, given that it has the

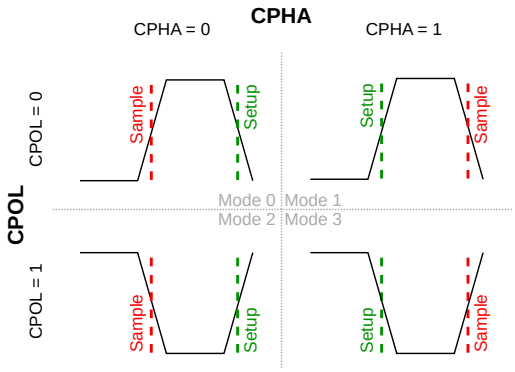


Figure 4.6: SPI modes and corresponding *CPOL* and *CPHA* settings.

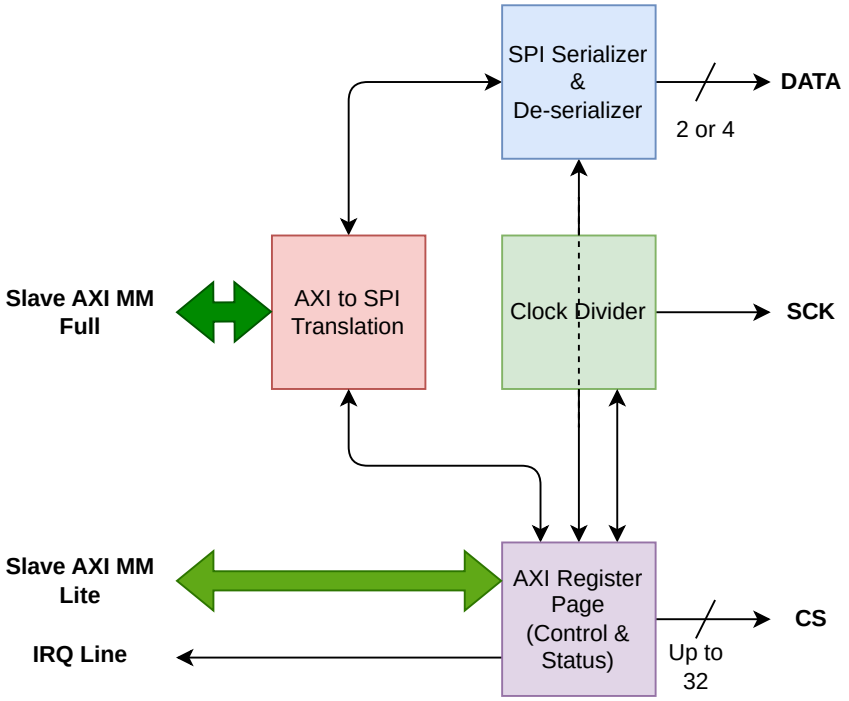


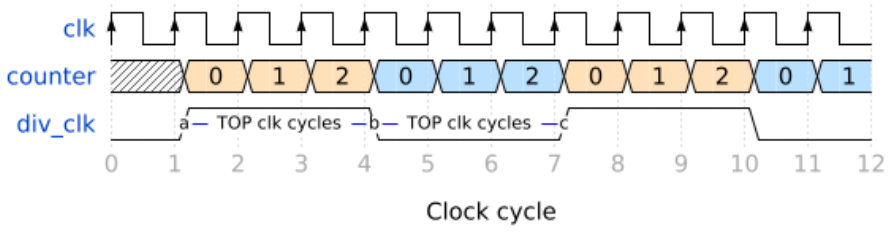
Figure 4.7: SPI engine IP core functional block diagram.

most features. All the synchronous logic in the core uses the same clock, so no special handling of clock domain crossing signals is required.

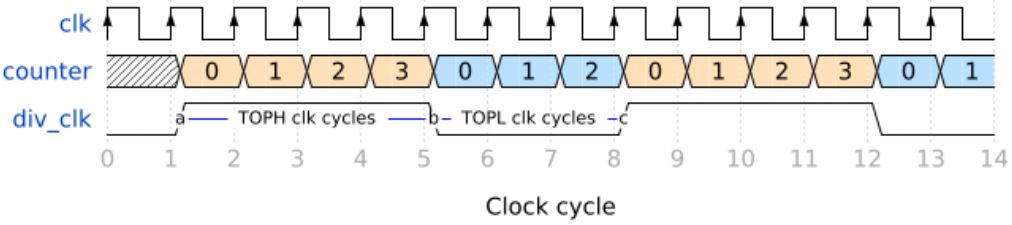
The core exposes three interfaces: an AXI4-Lite interface that offers register level access to configure and monitor the status of the core, and is documented at appendix B.2; an AXI4-Full interface for memory mapped accesses to the SPI flash; and, of course, the SPI interface itself, supporting multiple chip select lines.

The modularity principle, described in section 4.2.1 allows this core to re-use parts of the logic that were already described. The SPI interface clock is a divided down version of the module clock (AXI *aclk*), generated by a clock divider with a similar architecture as described in section 4.2.1. The big difference is that in the

SPI clock divider, the division ratio can be any integer value and is not limited to even integers. Odd division ratios are supported by generating asymmetric clocks which do not have a duty cycle of 50%. For this, the *TOP* value of the counter when the divided clock is 0 (*TOPL*) is different than the *TOP* value when it is 1 (*TOPH*), as can be seen in the timing diagrams of fig. 4.8. Figure 4.8a depicts the clock divider architecture employed in the I2S core, while fig. 4.8b represents the SPI clock divider. This increases flexibility at the cost of the aforementioned loss of



(a)



(b)

Figure 4.8: Clock divider module simplified waveforms: (a) Symmetric, $TOP = 2$; (b) Asymmetric, $TOPH = 3$ and $TOPL = 2$.

symmetry. If a 50% duty cycle clock is required for some reason, an even division ratio must be used. The interrupt request generation logic remains the same as the one described in section 4.2.1, although the relevant events that trigger interrupts differ. The AXI4-Lite memory mapped interface logic is also duplicated from the I2S core.

The SPI serialization and deserialization consists of a FSM which is governed by control signals or the AXI4-Full interface when in XIP (memory-mapped) mode. A simplified state diagram of this FSM is depicted in fig. 4.9. This machine accepts a main enable input (*spi_en*), which effectively serves as a synchronous reset, bringing everything to a known state. When enabled, the FSM sits in the initial state (*WAIT_XFER_REQ*) until either a read or write request is received. A write request always has priority over a read request, and is initiated by writing data to the shift-register output data buffer and asserting the signal *spi_sr_out_buf_valid*. A read request, on the other hand, is communicated to the FSM simply by asserting

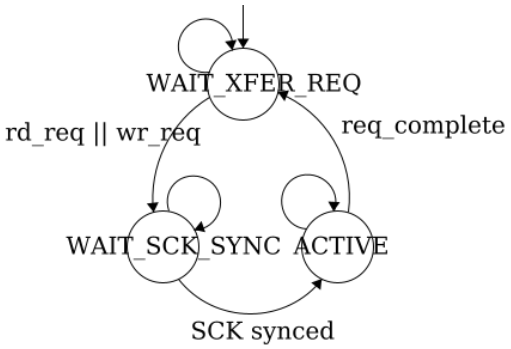


Figure 4.9: SPI serializer and deserializer FSM.

spi_rd_req. When either a read or write request is placed, the IO mode (count) for that request must be already set on *spi_io_mode*, so that the transaction happens in the desired mode, i.e. SPI, Dual SPI or QSPI.

The SPI clock signal (*SCK*) is free-running internally to the core, and only allowed to propagate outside during a transaction. Due to this behavior, once a request is made to the SPI FSM, the machine moves into a synchronization state (*WAIT_SCK_SYNC*), similar to the stages that synchronize *LRCLK* and *BCLK* in the I2S core (section 4.2.1). This stage synchronizes the machine to the sampling edge of *SCK*. This can either be a rising or falling edge, depending on the clock phase (CPHA) configuration of the core.

Once synchronized, the machine moves into the active transaction state, *ACTIVE*, where it takes actions only on the edges of *SCK*. As it was synchronized to the sampling edge before, the first edge to occur in this state is a setup edge. Here, the machine sets the data IOs direction (input or output) according to the received request and IO mode. If a write transaction was requested, the machine also shifts the correct amount of data (1, 2 or 4 bits) out of the shift registers and into the data IOs. On the sample edges of *SCK*, the machine moves data from the data IO lines into the shift register, if the transaction is a read. The machine works in byte mode, and the shift register has a length of 8 bits, so when the full byte is transferred, a new transaction can start. This may take eight, four or two clock cycles, depending on the configured IO mode. The core supports back-to-back transfers, so it does not need to go idle before subsequent requests. When a read transaction finishes, valid data is placed in the *spi_sr_in_buf* register and the signal *spi_sr_in_buf_valid* is asserted to notify interested parties that the aforementioned data is ready to be read.

As mentioned before, this SPI transaction state machine can be commanded manually, by writing individual bytes to specific registers and initiating transactions,

via the AXI-Lite interface, or it can be used to automatically translate transactions from the AXI-Full interface to SPI transactions. This automatic mode is configured and enabled via a set of bits on the register map of the core, but, once engaged, no further action is required. The logic that handles this translation is implemented via another FSM, which, by default, sits in the *READY* state, as depicted in the simplified state diagram of fig. 4.10. In this state, it asserts *arready* and waits for *arvalid*, signaling a pending read transaction request. Once *arready* and *arvalid* are

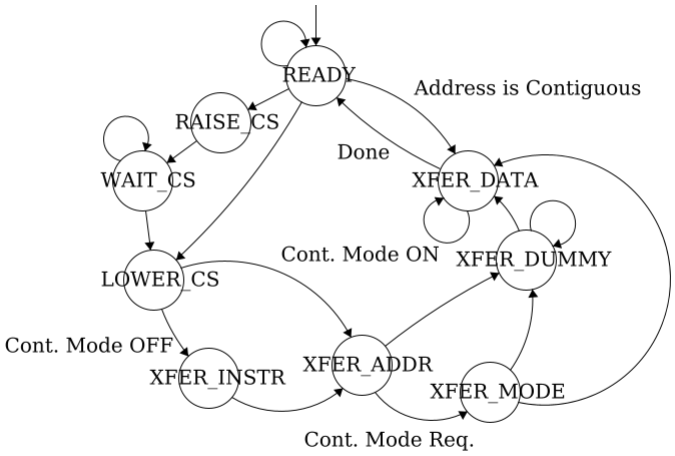


Figure 4.10: AXI MM to SPI FSM.

asserted, as per the AXI specification [64], a read address transaction takes place on the *araddr* bus. This address is stored, along with *arlen*, *arsize* and *arburst*. The core supports multi-burst transactions, depending on the *arlen* setting, as well different data widths, up to the data bus width (64-bits by default) via the *arsize* field. Additionally, unaligned addresses are supported, as required by the AXI spec. The core does not support simultaneous transactions. As soon as a transaction is received, *arready* is de-asserted, and is only asserted when the transaction finishes. The core is able to reduce read latency by using the continuous read mode, present on most SPI flash memories. This mode suppresses the instruction byte, sending the address directly, reducing the read latency by 8 *SCK* clock cycles. Another latency-reducing feature implemented by the core is the automatic identification of sequential read requests. When a subsequent read request arrives via the AXI bus, the core can detect when the address is contiguous to the last read address, and continue reading from the SPI memory without re-sending the address. This improves sequential read latency quite a bit. All this logic is implemented in the *READY* state, and ultimately determines the next state the machine will move to. If the address is contiguous, the machine skips most of the states, and moves directly to the *XFER_DATA* state. If the address is not contiguous, but the memory is already in continuous read mode, the machine pulses the *CS* line to restart the

transfer, so it can supply the new address to the flash. This is done via two states: *RAISE_CS* and *LOWER_CS* which simply control the *CS* line and wait for a configurable amount of clock cycles. If the flash is not yet in continuous read mode, which happens on the first ever AXI transaction after memory-mapped mode is enabled, the procedure to enable this mode is performed. The states *XFER_INSTR*, *XFER_ADDR*, *XFER_MODE* and *XFER_DUMMY* are all similar, in that they command the SPI FSM to transfer the proper data to the flash, to place it in the correct mode of operation. Once continuous read mode is enabled, only the *XFER_ADDR* state is required, as explained before. Independently of the state transitions that happen after the *READY* state, eventually, the machine will reach the *XFER_DATA* state. The amount of time (or clock cycles) that the machine spends in this state is entirely dependent on the length of the AXI transaction. Bytes are requested to the SPI FSM via the *spi_rd_req* signal, and data is streamed out via the AXI read data bus (*rdata*, *rvalid*, *rresp* and *rlast*). Once the transaction is completed, i.e., there is no more data to transfer, the machine moves back into the *READY* state. The write address, data and response channels of the AXI4-Full interface are present, however, the core does not currently support writing to the SPI bus in this manner. All write transactions received on the AXI bus are answered to with an *OKAY* response on *bresp*, but all data is discarded.

Three instances of the SPI IP core are used in this design. An SPI instance with no memory mapping and quad IO is dedicated to the RF transceiver, while a similar instance is dedicated to the mmWave synthesizer SPI interface.

The third instance is used to interface with the SPI flash, and is instantiated with the full feature set, which includes the AXI4-Full interface which directly maps the flash memory to the AXI address space, as explained above.

4.2.3 *AXI RF Timestamping*

The need for a timestamping specific core arises from the main limitation with most of the COTS market SDRs that utilize the reference HDL design from Analog Devices for the RF transceiver. That is, the lack of precise control over the time where a group of data samples should be transmitted, and, similarly, the fact that the receiver cannot be turned on or off at a specific point in time. This is a problem when the SDRs are used in time-division multiplexing (TDM) (or time-division duplex (TDD)) scenarios. This has sparked several discussions, and has impeded usage of some AD9361 based SDRs in some scenarios. Since the start of this work, however,

a couple of solutions to this problem have been developed. This timestamping IP core aims to be yet another solution to the problem. A comprehensive register map and additional documentation can be found at appendix B.3.

The core exposes a set of registers accessible via an AXI4-Lite interface, similar to the other IP cores presented, as shown in the simplified block diagram in fig. 4.11. The logic for the AXI interface is re-used from sections 4.2.1 and 4.2.2. There is a nuance, however, since this core exposes some 64-bit registers which must be read through the 32-bit data bus of the AXI-Lite interface. The correct read-out of data is achieved by buffering half of the 64-bit value when the other half is read. When reading the remaining bits via AXI, the buffered value is returned.

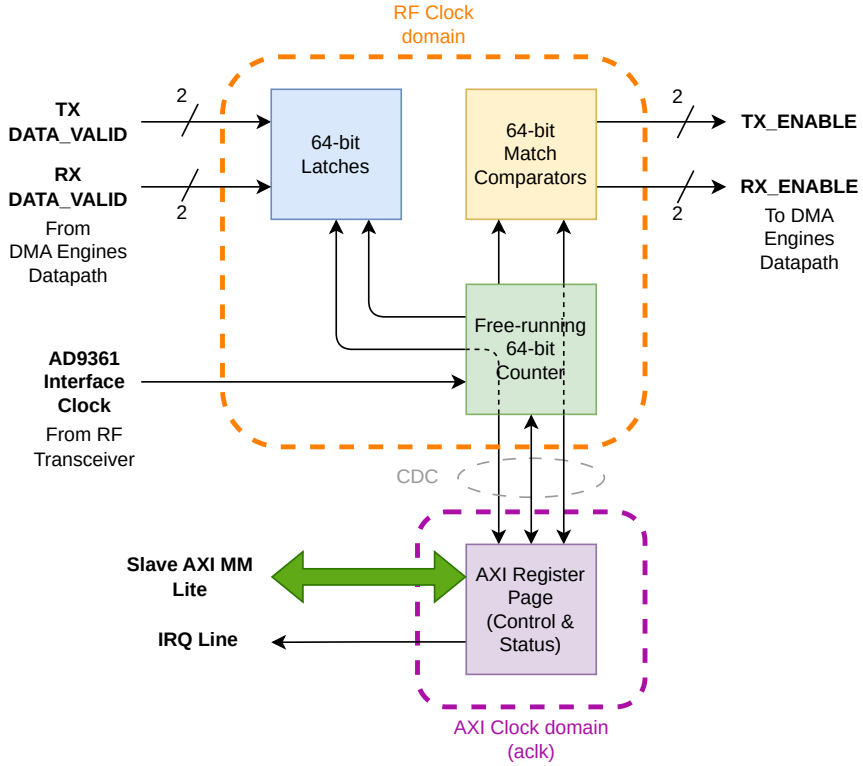


Figure 4.11: RF Timestamping IP core functional block diagram.

The core works by implementing a 64-bit counter that is clocked by the sample clock of the RF transceiver interface logic. Each clock cycle corresponds to a sample being transacted on the parallel bus, so this is used as a time base. On the receive side, there is a latch which can be armed on-demand. Once armed, the core looks for the first clock cycle where data transmission starts between the RF transceiver and the DMA engine, this happens when both the transceiver has valid data (*rx_data_ready* is asserted) and the DMA is ready to accept data (rx_dma_xfer_req is asserted). When this happens, the free-running counter value is latched and this value can be

read via the AXI bus later. This latched value is the time tag for the first sample that enters the DMA. Subsequent samples can be easily tagged by incrementing the latched value, as the received samples are contiguous.

Transmitter and receiver timed control are realized by programming the desired timestamp value via the AXI interface. The core constantly compares the current free-running counter value with the programmed timestamps and asserts a signal (either *tx_enable* or *rx_enable*) when a match occurs. These signals are used externally by the design to either apply back-pressure to the TX DMA to inhibit the transmitter operation or mask the data valid signal on the RX side to inhibit the RX DMA from capturing the data samples.

This IP also implements an interrupt request mechanism. Certain key events such as transmitter or receiver start and stop, data valid and overflow/underflow signals are multiplexed together on a single interrupt line that is output from the core. Interrupts can be individually masked and the interrupt source is readable via a register on the AXI interface.

As the architecture of this core involves logic clocked by two different clock signals, clock synchronization and crossing (CDC) measures are taken appropriately. All the signals that cross clock boundaries are synchronized with multiple flip-flops, in an attempt to minimize the probability of metastability. Data buses spanning multiple bits are synchronized with delayed handshake signals to prevent data corruption. As this core implements timing sensitive logic, it is architected in such a manner that the CDC logic does not interfere with the repeatability of timing events. An example of this is the comparison that takes place when enabling/disabling the transmitter or receiver. The comparison is always running on the sample clock, instead of on the AXI *aclk* domain.

4.2.4 *AXI IRQ Controller*

This custom IP core, whose internal block diagram is depicted in fig. 4.12, is used to manage several interrupt request lines originating from other cores or externally to the FPGA. It acts as a multiplexer and demultiplexer which allows routing of one or more input interrupt request line to an output interrupt line, for connection to a processor. The core also supports generation of PCIe message signaled interrupts (MSIs) from any interrupt source, when paired with a Xilinx AXI to PCIe IP core.

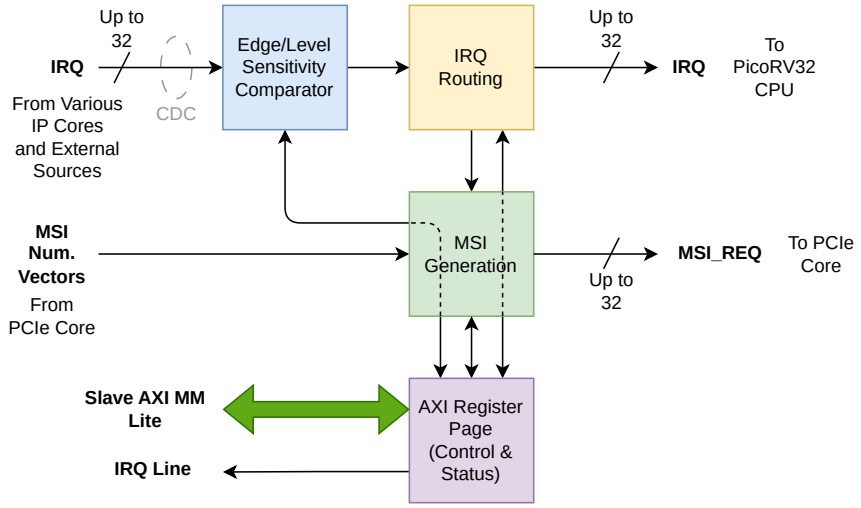


Figure 4.12: IRQ Controller IP core functional block diagram.

All the logic in the core is synchronous to a single clock, the AXI *aclk* clock. However, all the interrupt line inputs are treated as asynchronous, and thus go through a multi flip-flop synchronizer instead of being used directly. Edge detection is performed on every interrupt line by passing the signal through yet another flip-flop and comparing the data before and after the register. If they differ, then an edge is detected. This is the working principle of a simple edge detector.

The communication with the PCIe core is done thorough a couple of signals, as documented by Xilinx [65]. The *pcie_msi_enabled* signal reports to this core if MSI interrupts are supported at all, while the *pcie_msi_vector_width* signal a 3-bit wide vector which reports the number of available MSI interrupt vectors as negotiated with the PCIe root-complex (host). Both are used to validate requests for sending message signaled interrupts that are placed to the PCIe core during operation.

MSI requests are governed by a FSM which reads requests from a first-in first-out (FIFO) buffer. The FSM is initialized to the *IDLE* state, where it sits indefinitely until there is at least one entry on the FIFO. When this happens, the machine pops the valid FIFO entry and sets the *pcie_msi_vector* and *pcie_msi_request* signals accordingly. These signals are exposed out of the core and are also part of the interface with the PCIe core from Xilinx. The MSI request is considered accepted by the PCIe core when the signal *pcie_msi_granted* is asserted by it. Once this happens, the FSM goes back to the *IDLE* state to process further FIFO entries. This process is illustrated by the simplified state diagram shown in fig. 4.13.

The AXI interface logic is once again replicated from all the other cores, and exposes a couple of registers which allow reading the status of every interrupt line

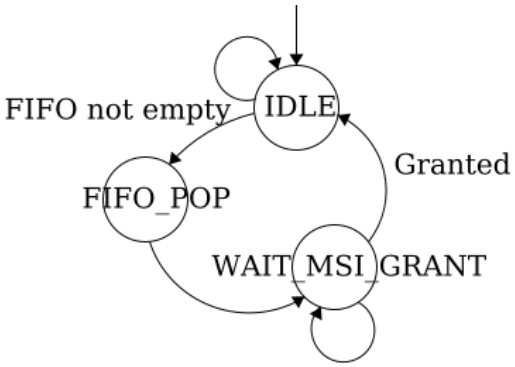


Figure 4.13: MSI handler FSM.

and allow an AXI master to clear a pending interrupt. Each input interrupt line has a dedicated register in which its configurations are accessible. It is possible to configure a line as rising edge, falling edge, high-level or low-level sensitive, and route it to any of the destinations available. The detailed register map and bitfield documentation is available at appendix B.4.

4.2.5 *AXI GPIO*

Although this may seem like a trivial piece of logic, Xilinx's solution does not quite accomplish some of the requirements introduced by the logic design associated with this work. The free of charge GPIO IP core provided by Xilinx with the Vivado suite allows the control of a variable number of GPIO lines via an AXI register interface, but has a functionality limitation which results in the host not being able to read the actual value of a GPIO line which is configured as an output. This was the main motivation for developing a replacement solution, as this is an important feature which facilitates coherency between software and hardware. When an hardware component is controlled with a GPIO and a software routine needs to check if said component is either enabled or disabled, it is crucial that the software has access to the real value present on the GPIO pin, instead of the desired value it previously set the pin to, since the real value on the pin can differ if an unexpected condition is impeding the pin from achieving the commanded logic level externally.

The logic that makes up this IP core is quite simple, and is depicted in fig. 4.14. The AXI register interface is carried over from other cores, therefore, it is not described here in detail to avoid redundancy. There are three main registers associated with GPIO functionality: A register to configure the GPIO direction, either input or output; a register to set the output value of each GPIO line; and finally, a register

to read the input data that's actually present on a particular GPIO pin. The input data is considered asynchronous and thus qualified using a multi flip-flop synchronizer. Additionally, four extra registers are included to facilitate atomic software interaction with the GPIOs. A write-one-to-set register is present for both the GPIO direction and output value. This register behaves exactly as it name implies. It it a write-only register, and any bit that is written to it as one results in a bit being set in either the GPIO direction or output data register vector. The complementary operation, write-one-to-clear, is also implemented through a different register. These special registers allow atomic accesses to be performed by software, avoiding the need for read-modify-write operations which may require resource locking. The full register map is documented in appendix B.5.

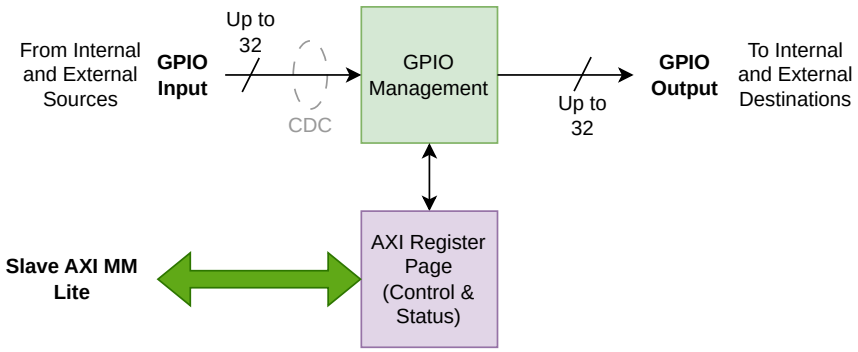


Figure 4.14: GPIO IP core functional block diagram.

Three instances of the GPIO IP core are used in the design. A GPIO instance is dedicated to the RF transceiver general purpose control lines, such as the mappable control and status signal buses, which expose several internal transceiver signals on dedicated I/O lines, and associated DSP logic. Overcurrent flags for the RF output amplifiers and bias-Ts are also available through this GPIO controller, as well as an auxiliary reset signal for the entire FPGA RF processing logic. Another GPIO core is also allocated to the mmWave synthesizer, used for the control and status lines, such as a global mute, chip enable, reset and lock detection. The last GPIO IP core is used for general purpose system control lines, such as auxiliary resets for the various clock domains of the design, a PCIe wakeup signal, a DDR3 interface calibration status indicator, a global clock enable for the clock management IC (the Si5351) and a PMBus enable signal.

4.3 PROPRIETARY IP CORES

The FPGA architecture presented in fig. 4.1 uses several proprietary IP cores, as listed in table 4.1. These cores play crucial roles in the operation and performance of the SDR base platform. This section describes key aspects of the proprietary IP, focusing on their implementation and functionality. It begins with an examination of the PicoRV32, a compact RISC-V CPU core known for its efficiency and flexibility [61]. The discussion then moves to the DMA controller IP, which facilitates high-speed data transfers, and the RF transceiver interface, essential for wireless communication. Additionally, the section covers the I2C interface IP, a widely used protocol for inter-chip communication, and concludes with an exploration of the PCIe to AXI bridge, which enables seamless integration between the PCIe bus and AXI-based systems. These proprietary IP cores represent key building blocks that contribute to the overall capabilities and performance of the SDR.

4.3.1 *PicoRV32*

The PicoRV32 core, developed by Clifford Wolf [61], is a compact and efficient implementation of the RISC-V instruction set architecture (ISA). It is designed to be small, portable, and highly configurable, making it ideal for FPGA-based applications and embedded systems where resource constraints are a primary concern. Unlike more complex RISC-V cores, PicoRV32 prioritizes area efficiency over raw performance, making it an excellent choice for applications that require minimal resource usage while still benefiting from the flexibility and extensibility of the RISC-V ecosystem. The entire core is implemented in a single Verilog file and also includes glue logic for connection to AXI networks, which facilitates integration in the Vivado workspace.

4.3.2 *DMA Controller*

The Analog Devices AXI DMA Controller IP [66] is a high-performance DMA engine designed for efficient data movement without CPU intervention. It facilitates high-speed data transfer between memory and peripherals using AXI memory-mapped, AXI-stream and Analog Devices' proprietary FIFO interfaces. This makes it particularly useful in applications requiring low-latency, high-throughput data

processing, such as this SDR. Unlike traditional CPU-driven data transfer methods, the AXI DMA operates autonomously, reducing processor overhead and improving system performance.

One of the key features of the Analog Devices AXI DMA Controller is its support for scatter-gather mode. This allows it to handle non-contiguous memory buffers efficiently, reducing the need for large, continuous memory allocations. Additionally, the DMA engine supports interrupt-driven operation, which notifies the system upon transfer completion or errors, enabling responsive software control. With configurable data widths and optimized support for high-speed ADCs, DACs, and transceivers (such as the AD9361 utilized in this SDR), the DMA controller is well-suited for the high-performance signal processing environments this SDR is designed for.

The AXI stream to AXI memory-mapped translation capability allows seamless integration between streaming data interfaces and addressable system memory. This is essential in multiple blocks of this logic design where continuous high-speed data must be efficiently moved, such as when interfacing with the RF transceiver and the audio CODEC. Furthermore, the IP is tightly integrated with Analog Devices' data converter solutions, ensuring optimal performance when paired with their high-speed transceivers.

This design encompasses six instances of this DMA controller. One per receive and transmit channel of the RF transceiver, one for sending data to the audio CODEC via the I2S interface and another one for receiving audio samples via the same interface. Although a simpler architecture could be achieved by, for example, using a single DMA controller for both transceiver receive and transmit channels, this slightly more complex architecture allows easier segmentation at the firmware level, which also simplifies software development in an application where separate applications may use different channels of the RF transceiver. Each DMA controller IP comprises an AXI-Lite interface which allows access to several configuration registers, as can be seen in fig. 4.2b.

4.3.3 *RF Transceiver interface*

The AXI AD9361 IP core [67] from Analog Devices is designed for integration with FPGA platforms to interface with the AD9361, AD9361 and AD9364 RF transceivers. The core comprises AXI-Stream like interfaces and performs the translation to the transceiver native parallel bus, allowing seamless data transfer between the

FPGA fabric and the transceiver, ensuring efficient high-speed communication. It supports both transmit and receive paths, handling I/Q data streams while managing synchronization, clocking, and control signals necessary for proper operation. Also included in the core are several advanced features to enhance signal integrity and system calibration. It integrates a pseudo-noise (PN) pattern generator and monitor to aid in the interface calibration process and data integrity testing, ensuring the system can perform self-calibration to achieve reliable communication between the FPGA and the RF transceiver. As shown in the internal core block diagram of fig. 4.15, the core also implements a digital direct synthesizer (DDS), enabling the generation of test waveforms such as sine and cosine signals for system validation and debugging. Additionally, it includes a DC removal block, which helps eliminate DC offsets in received signals, improving dynamic range and signal accuracy, especially in direct-conversion architectures. Also of particular importance is the included I/Q correction capability, which, when combined with the equalization features of the RF transceiver IC, further improve signal quality.

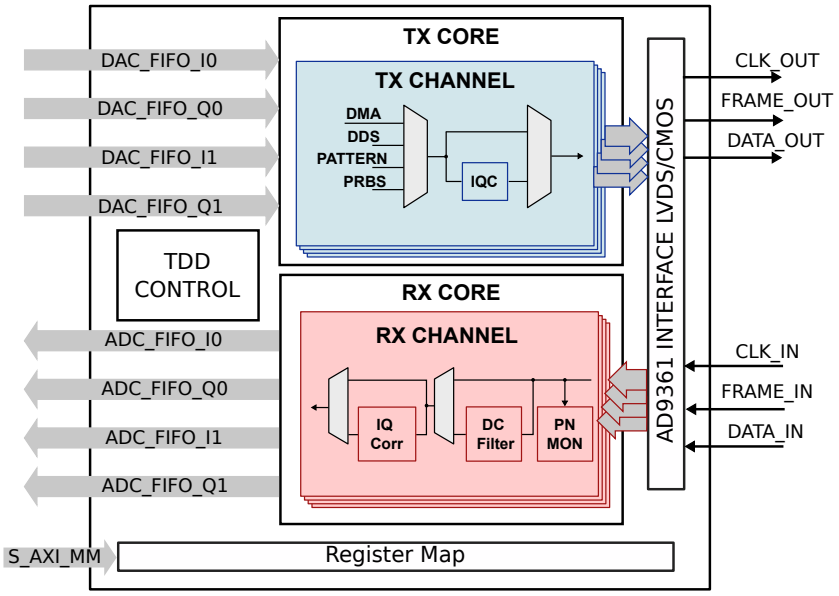


Figure 4.15: AXI AD9361 interface IP core [67].

The core includes an AXI-Lite control interface for configuration of the aforementioned features, enabling users to program settings such as DDS frequency, gain, and digital baseband correction coefficients. It also integrates tightly with the DMA engines, ensuring high-throughput data movement to and from memory.

The previously presented AD9361 and DMA IPs form the core of the RF signal processing chain. Together with data packers and unpackers from Analog Devices

and the RF Timestamping IP core developed in section 4.2.3, the I and Q data samples are moved throughout the system.

The ADC data packer and DAC data unpacker IP cores [68] from Analog Devices provide a way of combining multiple vectors of data into a single one, and vice versa. This facilitates the interface with the fixed width bus of the DMA controllers. As the RF transceiver interface IP core outputs up to 4 16-bit words (TX0 I, TX0 Q, TX1 I, TX1 Q), which may not be valid at the same time (depending on the number of channels being used), the packer IP can aggregate the active channels into a single, wider bus, discarding data from invalid/inactive channels. Figure 4.16 exemplifies the working behavior of the packer core under different operation scenarios. In this example the core is configured with four 32-bit inputs (A through D) and a single 128-bit output. When the system signals the core that the four inputs contain valid

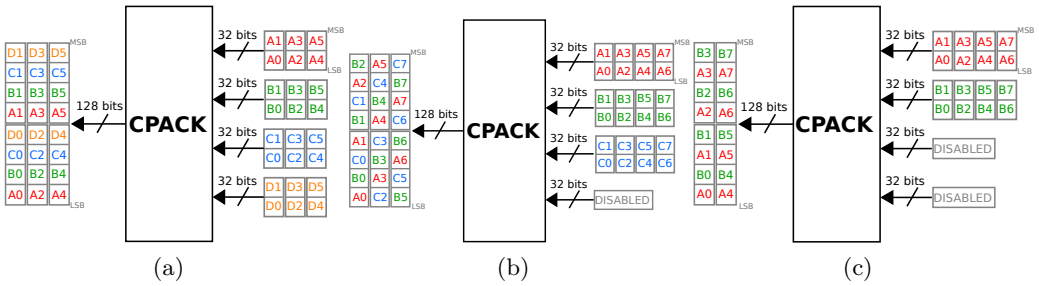
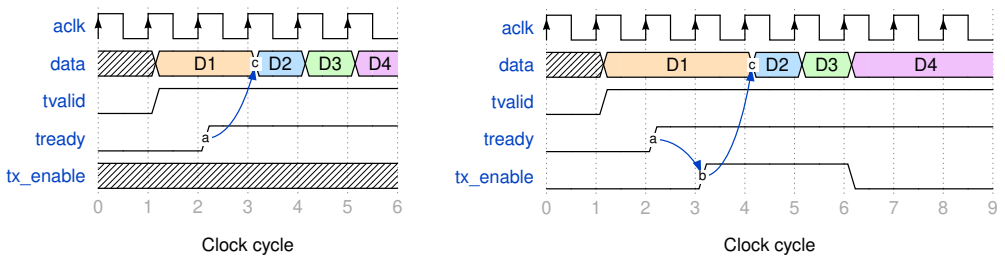


Figure 4.16: Analog Devices data packer [68] behavior under different conditions: (a) 4 channels enabled; (b) 3 channels enabled; (c) 2 channels enabled.

data, the core packs each input sequentially into the output, as seen in fig. 4.16a. During operation, if the D input is disabled, the core can automatically adapt, omitting the D channel from the output, as depicted in fig. 4.16b. Figure 4.16c exemplifies the behavior in a scenario where two of the channels do not contain valid data.

The RF Timestamping core is integrated into the transceiver datapath by applying backpressure to the transmit and receive DMA engines via the *tx_enable* and *rx_enable* signals mentioned in section 4.2.3. An example of this working principle is depicted in fig. 4.17. In fig. 4.17a, an AXI-Stream transaction occurs normally, when the transmitter is not being gated by the timestamping core. Every *aclk* clock cycle where *tvalid* and *tready* are simultaneously asserted, a data transfer takes place. In fig. 4.17b, the AXI-Stream rules still apply, but with an addition. A data transaction can only happen when the *tready*, *tvalid* and *tx_enable* are all asserted.



(a) (b)

Figure 4.17: TX gater.

4.3.4 *I2C interface*

A simple I2C core provided by Xilinx is used to interface with the PMU and clock management IC, while a second instance of this core connects to two of the expansion card interface I/O lines, for communications with installed expansion cards. A third instance is used to interface with the audio CODEC I2C control interface, while an I2S core (described in section 4.2.1) is used for the CODEC media interface.

4.3.5 *PCIe to AXI bridge*

The Xilinx AXI to PCIe IP core serves as a high-speed interface between the FPGA logic design and any PCIe host. It acts as a bridge between the AXI memory-mapped bus and PCIe Gen 2.0, enabling efficient data transfer with minimal latency. One of the core's main strengths is its seamless integration with AXI-based logic within the FPGA. By converting AXI transactions into PCIe transaction layer packets (TLPs), it ensures smooth communication between the host system and the FPGA fabric. It provides scalable throughput by supporting multiple PCIe lane configurations (x1, x2, x4 and x8), although the available options depend on hardware support.

In addition to high-performance data handling, the IP offers PCIe base address register (BAR) support, allowing the FPGA to be mapped into the host system's memory space. This feature simplifies communication between the FPGA and CPU, abstracting address translation and facilitating low-latency read and write operations initiated by the CPU. It is also possible to negotiate bus master capabilities with the host system, and, when this is done, the FPGA gains direct access to a portion of system memory. This is a powerful feature which allows other AXI bus masters

inside of the FPGA to perform operations directly on the host system's memory, such as copying large amounts of data, without the host CPU intervention. Paired with PCIe message signaled interrupts, also supported by the core, the host can then be informed of the end of a data transfer (among other relevant events) so the CPU can further process it. When using this feature, the six upper regions depicted in fig. 4.2a become "windows" into the host system's memory, where the DMA controllers can be pointed at to either read or write chunks of data.

4.3.6 *DDR3 Memory Interface*

The Xilinx memory interface generator IP is a crucial component for interfacing the DDR3 memory with the FPGA. This IP simplifies DDR3 memory controller implementation by providing a pre-verified, configurable interface that meets JEDEC standards. It supports multiple memory types, including DDR3 and (LP)DDR2 and provides essential features like automatic timing calibration, command scheduling, and error detection. It can be configured to work in different memory access modes, including burst-based transactions and low-latency operations for performance optimization. The IP supports multiple clocking modes, such as a single-ended system clock or a differential clock to enhance signal integrity, while integrating calibration logic to automatically adjust delays and ensure signal alignment, which is crucial for achieving high-speed operation.

For ease of integration into AXI-based designs such as the SDR base platform, the IP offers an AXI4 interface, allowing seamless connection with the AXI interconnect. Burst length settings, read/write latency adjustments, and refresh interval configurations are fine-tuned in the graphical interface in Vivado to optimize performance for the specific workloads involved in the SDR.

A key aspect of MIG configuration is clocking. The MIG IP uses a memory controller clock derived from the synchronization unit to generate the necessary phase-aligned clocks for DDR3 read/write operations. The clocking scheme involves a clock used for the physical layer and another clock for the AXI interface.

4.4 SOFTWARE

The software infrastructure is a fundamental component of the developed software defined radio platform, serving as the critical interface between the custom FPGA

logic, the designed PCB, and higher-level user applications. It orchestrates the configuration, control, and data flow management, ensuring seamless integration of the hardware elements while enabling flexible and efficient signal processing. This software stack encompasses a low-level kernel driver, MCU firmware, and higher-level control frameworks, effectively acting as the cohesive layer that binds the system's heterogeneous components discussed so far into a unified, functional platform. Its robustness and optimization directly impact the SDR's performance, adaptability, and scalability, making it an essential aspect of the overall design.

4.4.1 *Power management controller firmware*

The power management controller firmware is the brain of the PMU, as it governs the behavior of the SDR even before any of the signal processing components power up. Thus, it plays an important role on the operation of the SDR.

The firmware for the ARM Cortex-M0+ micro-controller unit is developed in the C programming language. In section 3.2.5, the implemented algorithm has already been briefly mentioned. On boot, the MCU initiates all the peripherals, including communication interfaces such as the I2C interface, which is firstly configured as a master. Via the I2C interface, the pre-regulator is initialized and the desired operation parameters are programmed into it, such as the output voltage, current limits and switching frequency. Following that, all three main power sources are checked using the ADC, and the highest level supply is selected to power the system. When no power source meets the specified criteria (e.g., voltage greater than 7 V), the PMC uses an LED to warn the user, and keeps checking all the sources until at one is available.

When a power source is ultimately selected and connected to the system, the pre-regulator is then enabled, which triggers the entire power tree as per the sequence illustrated in fig. 3.9, and keeps monitoring the pre-regulator via its digital interface to ensure it starts properly and no faults occur. After this is done, the PMC no longer needs to behave as an I2C master, and is instead required to present itself as a slave on the bus, so that the FPGA can communicate with it. Because of this, the I2C interface is re-initialized as a slave. At this point, the entire system power rails are available, and the FPGA reset line is released, so that the device can initialize from the SPI flash memory, and the CM4 enable signal is asserted, allowing the operating system to boot, completing the power up sequence.

4.4.2 *Addressing and kernel device driver*

The majority of the host system's operations which interact with the SDR platform hardware can be simplified to read and/or write operations, which are targeted at the AXI address space on the FPGA. Everything from writing to a control register of an IP core to moving large amounts of data (e.g., I/Q samples) can be simplified to a read or write operation. Naturally, an operation of this kind requires an address, which is either being written to, or read from. Due to the complexity of such a system, however, several address domains exist, and between each domain there is a block that performs address translation. Figure 4.18 illustrates, at a high (block) level, this addressing scheme, while fig. 4.19 details the address translations that may occur, with the block responsible for the translations. The PCIe IP core, described in

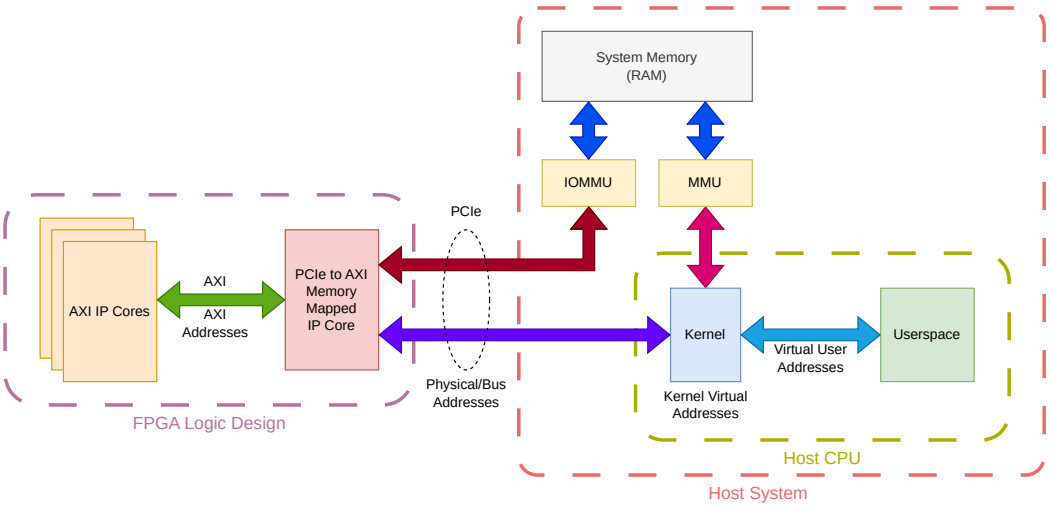


Figure 4.18: System address spaces summary.

section 4.3, can be seen in figs. 4.18 and 4.19 performing the translation between the AXI bus, which is completely enclosed in the FPGA, and the PCIe bus (i.e., PCIe TLPs), which addresses devices via a physical address. The Linux kernel, which runs on the host CPU, as depicted in fig. 4.18, mainly uses kernel virtual addresses (i.e., in memory allocated via *kmalloc*, a kernel memory allocation method), and is responsible for properly translating addresses when accessing external physical devices (e.g., on the PCIe bus). To enforce security and process isolation, Linux has separate virtual address regions: the kernel space and user space [69]. For this reason, another address translation is needed when userspace processes need to access, for example, the AXI bus on the FPGA. This conversion is performed by the Linux kernel itself. The piece of kernel software responsible for interfacing with

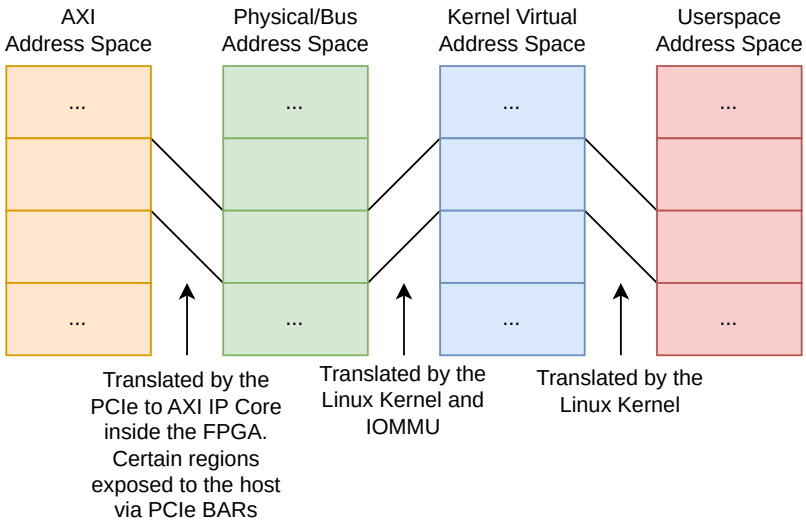


Figure 4.19: Translation between system address spaces.

physical system devices is the kernel device driver [70], which must be loaded onto the host system in which the SDR is installed.

The developed Linux kernel driver provides an interface between the SDR FPGA and the host system via PCIe. It is designed to manage the FPGA's resources by enabling efficient communication, including register access, memory mapping, and DMA operations. As part of the initialization process, the driver registers a PCIe device that matches the vendor and device identifiers programmed into the PCIe IP core. When a matching device is detected in the system, it sets up memory regions using BARs and registers an interrupt handler in order to support event-driven operations [70].

The implementation includes DMA buffer management functions that allow dynamic allocation, deallocation, and querying of coherent memory buffers, which are essential for high-speed data transfer between the host and FPGA. These buffers ensure efficient memory access with minimal CPU overhead. The driver also provides *ioctl*-based control interfaces, allowing userspace applications to request specific FPGA operations, such as memory mapping, buffer allocation, and hardware queries like retrieving the device's serial number or design version. Additionally, the driver handles interrupts using MSIs to notify the host about FPGA events, enhancing performance by reducing polling overhead [70].

To ensure seamless integration with the Linux kernel and operating system in general, the driver registers itself as a character device, making it accessible via */dev/icyradioX*. This allows applications (userspace processes) to interact with the SDR through standard file operations like *read*, *write*, *mmap*, and *poll* [70]. The

memory-mapped interface enables userspace applications, such as the SoapySDR [16] support library, addressed in the next section, to directly access the AXI memory space inside the FPGA, and thus make use of the SDR platform.

4.4.3 *Userspace SoapySDR driver*

SoapySDR is an open-source software defined radio abstraction layer that provides a unified application programming interface (API) for interfacing with different SDR hardware [16]. It serves as a hardware abstraction library, allowing applications to interact with multiple SDR devices using a consistent API, regardless of the underlying hardware differences. SoapySDR supports a wide range of COTS SDR platforms, including LimeSDR, HackRF, RTL-SDR, USRP, and ADALM-Pluto, making it a flexible solution for cross-platform SDR development. It also integrates well with popular SDR frameworks such as GNU Radio [14], Pothos [71], and SDRangel [72], enabling seamless signal processing and communication system development. By providing a standard API for device discovery, configuration, streaming, and tuning, SoapySDR simplifies SDR application development, allowing users to focus on high-level functionality rather than low-level hardware-specific details. Building an SDR platform which integrates with the SoapySDR framework is key to achieve high adoption of the developed SDR platform, due to the ease of transition from other existing SDRs.

The SoapySDR support library developed for the SDR presented in this work is primarily developed using C++, aligning with the language choice of the core SoapySDR framework. This selection leverages C++'s object-oriented features to create a modular and extensible codebase, facilitating seamless integration with the hardware that supports the SDR.

C++'s classes and inheritance are leveraged to achieve a comprehensive and organized software library. Each block of the high level block diagrams of figs. 3.1 and 4.1 correspond to a C++ class, with proper inheritance and polymorphism usage. A particularly useful example of this is the AXI bus peripherals. A base class is created with simple register read and write methods, as well as properties such as base address within the memory map. This base class is abstract and, therefore, not instantiable. All the AXI bus slaves descend from this class and inherit its methods and properties, further implementing more specific functions that are related with the core's functions.

The object-oriented features of the language are also extensively used throughout the library source code. A device on an I2C bus, for example, is implemented in its own class, whose constructor method receives an I2C controller and slave address as parameters. Besides fostering organization, this architecture also improves code portability. Continuing the I2C slave example, this allows two or more instances of the same slave device class to be present on different buses, or the same bus with different slave addresses.

The SoapySDR driver also extensively employs multithreading to split the computational load between multiple threads, which, with modern multi-core CPUs, can be ran simultaneously. Multithreading is a programming paradigm that allows concurrent execution of multiple threads within a single process, enabling improved performance, especially in multi-core processors [73]. The C++11 standard introduced the *<thread>* library, which is employed in the driver as a standardized way to create and manage threads. However, with parallel execution comes the challenge of data synchronization. When multiple threads access shared resources, issues like race conditions and data corruption can occur. To prevent such problems, C++ provides synchronization primitives like mutexes (mutual exclusion) and atomic variables, which are broadly utilized throughout the library.

A mutex ensures that only one thread at a time can access a shared resource by locking and unlocking access points. Alternatively, atomic variables provide lock-free synchronization mechanisms, ensuring that operations on shared data are performed as atomic (i.e., non-interruptable) transactions. They are especially useful in low-latency applications where mutex overhead is undesirable, such as in the driver portions which handle real-time data streaming, e.g., from the RF transceiver [73].

The developed SoapySDR support library interacts with the kernel device driver through system calls. Since the SoapySDR support library runs in userspace, which is a restricted environment with limited access to hardware resources, it must use system calls to request services from the kernel, such as reading from or writing to AXI slaves residing on the FPGA. The kernel device driver, addressed in section 4.4.2, exposes its functionality to user-space through system calls like *open*, *read*, *write*, *ioctl*, and *mmap*. These system calls act as an interface, allowing controlled access to hardware while ensuring system stability and security [70].

The character device file created and registered by the driver (*/dev/icyradioX*) represents the physical device. When a user-space program opens a device file using *open*, the kernel routes the call to the device driver's *open* function. Similarly,

read and *write* system calls are mapped to driver-specific functions that handle data transfer between the user application and the hardware. For more complex interactions, such as interrupt notification and handling, *ioctl* (input/output control) provides a flexible mechanism to send commands or retrieve device-specific information that does not fit into simple read/write operations. Additionally, *mmap* is used by the SoapySDR support library to map device memory into user-space, allowing high-performance direct memory access, e.g., for data streaming [70].

4.4.4 *User applications*

Alongside all the software packages which are crucial for the operation of the SDR, an usage example application were also developed, to test and demonstrate the flexibility of the developed SDR. This application is not a required element, but a base that exemplifies the usage of the SoapySDR API to implement a transmitter and receiver DSP chain, with the goal of transmitting and receiving a RF signal modulated with a live video feed. Among the SoapySDR library, for interfacing with SDR hardware, the Liquid DSP library [15] is also used to implement the core of the signal processing chain.

Liquid DSP is a lightweight, open-source digital signal processing library designed for SDR and wireless communications. It provides a broad collection of DSP utilities, including signal filtering, modulation and demodulation, error correction, equalization, and spectral analysis. Written in C, the library is optimized for performance and portability, making it well-suited for use in embedded systems, real-time SDR applications, and experimental wireless designs. A key feature of this library is its modular architecture, which allows developers to integrate specific DSP functions as needed. It supports a variety of modulation techniques (such as amplitude, frequency and phase modulation, M-QAM, and OFDM) and includes efficient implementations of FIR and IIR filters for signal conditioning. Its lightweight nature, combined with minimal dependencies, makes it an excellent choice for custom SDR development, wireless communication research, and real-time signal processing applications [15]. The performance of this library is further boosted by usage of SIMD implementation of core DSP algorithms.

The transmitter application, whose architecture is depicted in fig. 4.20, is designed to be executed on an SDR with a CM4 installed, given that it takes advantage of the hardware accelerated video encoders present on the BCM2711 SoC. It is,

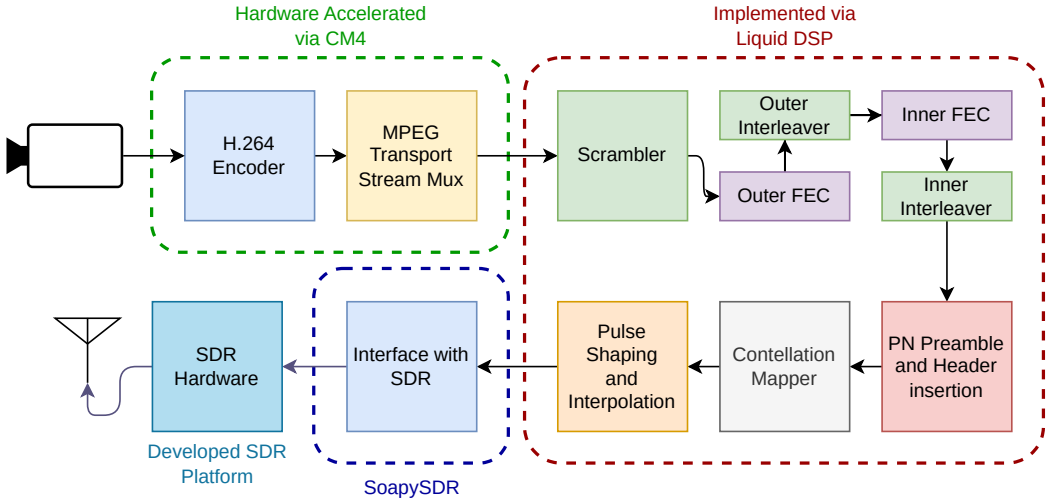


Figure 4.20: Demonstration streaming application transmitter simplified block diagram.

however, not limited to this use, since it's usage is also possible on an SDR which is installed on a PCIe slot of another host platform.

The transmitter captures live raw video frames from a camera, attached via CSI to the CM4, performs H.264 encoding to reduce the required bitrate and packs the stream into a Moving Picture Experts Group (MPEG) transport stream (TS), for transmission over the radio link. Despite being an older CODEC, H.264 is employed here due to the included hardware acceleration support of the BCM2711 SoC. Being an older standard, it is also less computationally intensive, despite achieving lower encoding efficiency (i.e., image quality vs bitrate).

The MPEG transport stream is well-suited for this application due to its robustness, error resilience, and efficient multiplexing capabilities. Designed for real-time streaming, the TS divides data into fixed-size packets (188 bytes) that include synchronization markers, making it more tolerant to packet loss and suitable for environments with high bit error rates, such as this wireless radio link. Additionally, MPEG transport streams supports seamless multiplexing of multiple audio, video, and metadata streams, ensuring efficient bandwidth utilization. The format's flexibility allows for adaptive bitrate streaming, and is capable of inserting filler packets to match a constant bitrate, making it ideal for broadcasting applications where maintaining signal integrity over varying network conditions is critical [74].

TS packets are then piped to the modulator application, which leverages several Liquid DSP functions which scramble the data according to a repeatable pattern (whitening), add redundancy (FEC) to the data, modulate the signal onto a carrier and limit the signal bandwidth (pulse-shaping). Additional information is also

appended to the TS bitstream, which helps a receiver to synchronize to the start of each frame. A PN sequence is used for this effect.

PN sequences are integral to frame synchronization in this streaming radio link, primarily due to their distinctive correlation properties. A key characteristic of PN sequences is their impulse-like autocorrelation function, which exhibits a sharp peak at zero lag and minimal values elsewhere. This sharp autocorrelation peak enables receivers to precisely identify the start of a frame by correlating the incoming signal with a locally generated PN sequence; the pronounced peak indicates accurate alignment. Additionally, PN sequences are designed to have low cross-correlation with other sequences, minimizing the likelihood of false synchronization due to interference from other signals, ensuring reliable frame synchronization [75].

After the complete radio frames are assembled, the application then passes the raw I/Q samples to the SDR via the previously mentioned SoapySDR API.

The receiver application, illustrated in fig. 4.21, comprises many blocks analogous to the transmitter, despite performing the inverse functions. Raw I/Q data is

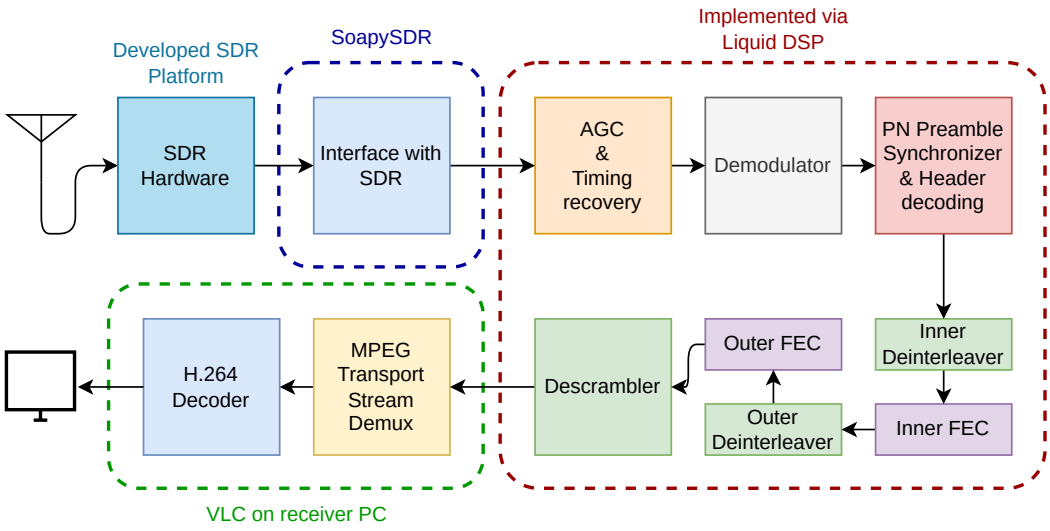


Figure 4.21: Demonstration streaming application receiver simplified block diagram.

received from the SDR, via the SoapySDR interface API, which is then processed by an automatic gain control (AGC) algorithm, which ensures a constant signal power, and a timing recovery mechanism, which re-samples the signal at the maximum symbol energy instant. After timing recovery, the symbols are demodulated and the receiver is synchronized to the start of frames using the aforementioned PN sequence. When a frame is detected, FEC decoding is performed to correct residual errors, and after de-scrambling the received payload, the raw TS frames are provided at

the application's output, which are then directed to a media player, which displays the received video stream on a computer monitor.

5

MMWAVE EXPANSION CARD

The mmWave expansion card is a critical component designed to enhance the capabilities of the SDR base platform by enabling operation at millimeter-wave frequencies. This extension opens-up new possibilities for high-bandwidth communications and sensing applications. This chapter offers a comprehensive overview of the mmWave expansion card design, focusing on key aspects such as system design, circuit design, performance estimation, and firmware/software integration. The chapter begins with an introduction to the mmWave card's functional architecture and system design considerations, including simulation approaches and the choice of PCB material stackup. It then explores the circuit design, emphasizing the design of functional blocks that ensure the stability and optimal performance of the overall mmWave system. The chapter also includes an assessment of the estimated performance of the mmWave board, discussing and evaluating several key performance parameters. Finally, the firmware and software necessary for monitoring and controlling the mmWave expansion card within the SDR platform are addressed.

5.1 SYSTEM DESIGN CONSIDERATIONS

To design an SDR expansion card circuit that meets the desired operation and satisfies the specifications outlined in section 2.6, it is first necessary to define the system topology and the design approach for each functional block. Additionally, in the RF domain, particularly at mmWave frequencies, circuit design must be supported by simulations that require a thorough understanding of the material properties used in the circuit board. The following subsections explore these essential topics, which are crucial for designing the circuit of each functional block.

5.1.1 *mmWave card functional architecture*

The architecture is based on two single-conversion stages, one for up-conversion and the other for down-conversion, as depicted in fig. 5.1. This figure shows a

comprehensive and simplified functional block diagram of the mmWave expansion card. The mmWave expansion card must have two IF ports, one for transmission and the other for reception, with a bandwidth of at least 500 MHz centered at 3 GHz. These IF ports are meant to interface directly to the RF ports of the SDR base platform, but can be connected to any other signal transceiver. The two mmWave RF interface ports, one for transmission and the other for reception, can be used to connect the mmWave expansion card to other devices (e.g., amplifiers and filters), equipment and directly to transmitting and receiving antennas. The mmWave expansion card must also have an internal local oscillator (LO) block and also allow the operation with an external LO by providing a dedicated port for the external signal. This architecture enables independent selection of LO source for both the up and downconverters, in order to support FDD operation. Moreover, this LO design approach not only introduces operational flexibility but also broadens its application range by enabling the use of external oscillators with very low phase noise—something that cannot be achieved with highly integrated Si-based wide tuning range variable-controlled oscillators. This enhancement maximizes the potential of the SDR in demanding performance applications, particularly those involving signals with high m-ary modulations, which are highly sensitive to LO phase impairments. The selection between the use of the internal or the external LO is performed in the LO selection blocks illustrated in fig. 5.1. The circuit design is based on the selection of existing components suitable to operate in the mmWave band that fulfill the desired performance. The PCB layout design considerations to ensure optimal performance at high frequencies is also approached and sustained by circuit-layout simulations.

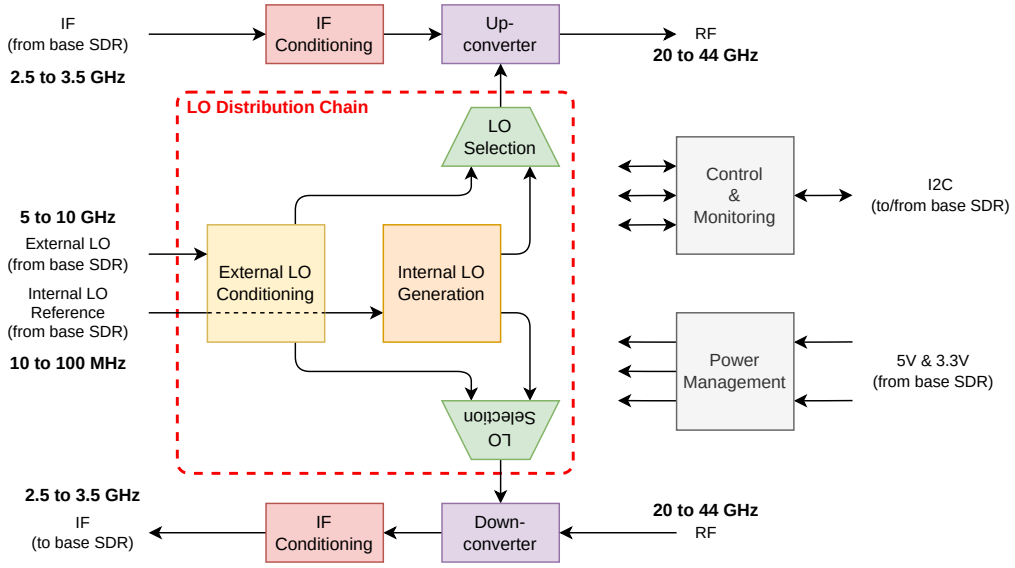


Figure 5.1: mmWave expansion card functional block diagram.

5.1.2 *Simulation considerations*

Both circuit (schematic level) and electromagnetic (layout level) simulations play a crucial role in the design and optimization of RF circuits before physical implementation on a printed circuit board, especially at mmWave frequencies, where even small layout variations can significantly impact performance. At these high frequencies, parasitic effects, transmission line discontinuities, and coupling between components become more pronounced, making accurate modeling crucial before fabrication. Electromagnetic simulations are key for estimating undesired circuit couplings that could lead to instability, such as oscillations in amplifiers caused by positive feedback. Additionally, to estimate and fine-tune the performance of certain functional devices developed directly on the PCB, such as directional couplers and power dividers (which may not have a schematic model), electromagnetic simulations are essential, as there is no other efficient method for evaluating their performance. However, it is important to note that, in order to assess the circuit's performance using both circuit and electromagnetic simulations, it is necessary to have access to the characteristics of the printed circuit board materials, including material dimensions (e.g., conductor and dielectric thickness) and electrical properties (such as conductor conductivity, dielectric permittivity, and the loss tangent of the dielectric).

Using simulation tools, it is possible to gain valuable insights into the overall circuit performance, identify potential issues, and refine designs without the costs and time associated with multiple fabrication iterations. During the design process of each functional block of the mmWave expansion card, both circuit and layout simulations were conducted to fine-tune and optimize performance. The tool used for this purpose was Cadence's AWR Microwave Office.

5.1.3 *PCB stackup*

Before laying all the components on the PCB and routing all the interconnecting traces, it is necessary to define the right material stacks suitable for mmWave and general purpose layouts, since the board has important RF, power, analog and digital parts all integrated together. It is also necessary to have a final decision on the stackup before performing any kind of electromagnetic (EM) simulations, since the material has to be defined in the simulation environment and plays a crucial role in the results. Several possibilities were considered, mainly a full FR-4 symmetric

stackup, full Rogers manufactured materials stackup and hybrid solutions. Rogers Corporation materials are well known in the industry for offering remarkable RF performance and have well-defined models available for electromagnetic simulations, but come at high cost. While the compromises for the base platform were tolerated in favor of a cheaper PCB material stackup, the frequencies involved in the mmWave expansion card are a order of magnitude higher than on the base SDR, and the shortcomings of using FR-4 at these frequencies cannot be tolerated. No blind and/or buried vias were considered, since they add quite a lot of manufacturing cost and, although they can be though of as a nice-to-have feature, they are not a strict requirement in this application.

Using the custom hybrid stack-up, depicted in table 5.1, with RO4350B between the top layer and the first internal layer, while employing FR-4 at the remaining layers, offers a balance between RF performance and cost efficiency in the mmWave expansion card. The choice of RO4350B between the upper layers ensures a controlled and stable dielectric environment for high-frequency signals. With a typical dielectric constant of 3.48 and a low dissipation factor ($\tan\delta$) of around 0.0037 at 10 GHz, this material minimizes losses and phase variations, which are critical at mmWave frequencies.

Table 5.1: mmWave expansion card PCB stackup.

Layer	**Material Type**	**Thickness**
Top	Copper	40.6 μm
	RO4350B	0.254 mm
Layer 2	Copper	17.8 μm
	Prepeg 7628	0.5328 mm
Layer 3	Copper	17.8 μm
	FR-4 Core	0.61 mm
Bottom	Copper	40.6 μm

By placing RO4350B between the top signal layer and the first internal ground plane, microstrip or grounded co-planar waveguide (CPWG) structures can maintain consistent impedance and low insertion loss, crucial for preserving signal integrity. Meanwhile, using FR-4 in the lower layers reduces manufacturing costs. FR-4 can be tolerated in this situation, since no high-frequency signals are routed on those layers.

5.2 CIRCUIT DESIGN

This section discusses the design process of the electronic circuits and parts of the PCB for the mmWave expansion card, which is intended to operate within the 20 GHz to 40 GHz frequency range, as outlined in section 2.6. It also presents the simulation results for the designed RF circuit, focusing on key performance parameters such as return loss, insertion loss/gain, and stability. These simulations provide valuable insights to ensure that the final PCB implementation meets the required specifications, with minimal need for design adjustments. Due to the complexity of the full signal chain, the design is divided into individual blocks that are simulated separately.

5.2.1 *Frequency conversion*

The mmWave expansion card is designed around two main integrated circuits from Analog Devices, the ADMV1013 [43] and ADMV1014 [44]. These are up and downconverters, respectively, working in the 24 GHz to 44 GHz RF range. At the time of design, these ICs represent the most integrated solution available, capable of operating over such a wide frequency band, while maintaining good operating performance, both appealing characteristics for use in software defined radios.

The ADMV1013 upconverter supports two frequency translation modes: direct RF conversion from baseband IQ input signals and single sideband (SSB) upconversion from quadrature (complex) intermediate frequency inputs. For this application, the baseband input is disabled and left disconnected, enabling the insertion of modulated complex IF signals ranging from 0.8 GHz to 6.0 GHz. These signals are upconverted to a central frequency between 24 GHz and 44 GHz range by mixing with an appropriate LO signal, conditioned on-chip. A divided (by 4) version of the LO signal is supplied to the chip, which internally multiplies, filters, amplifies and generates quadrature as required, before feeding the internal mixers. This divided LO signal must have a power level that's between -6 dBm and 6 dBm, to ensure correct operation of the device. As such, the LO conditioning and distribution chain of fig. 5.1 is designed to comply with those specifications.

The IC comprises a digital communication interface which allows quadrature phase and mixer gate voltage adjustments to optimize sideband suppression and local oscillator leakage nulling. The IC also contains programmable RF amplifiers, and a power/envelope detector, which gives control over the RF output power of

the device. A simplified internal block diagram of the ADMV1013 is depicted in fig. 5.2.

The ADMV1014 downconverter, shown in fig. 5.3, performs the inverse function of the ADMV1013. These two ICs share similar internal blocks. The LO conditioning is done in the same way as the upconverter, and the downconverter also supports both direct-conversion and IF modes. As with the upconverter, the present expansion card employs the IF operation mode.

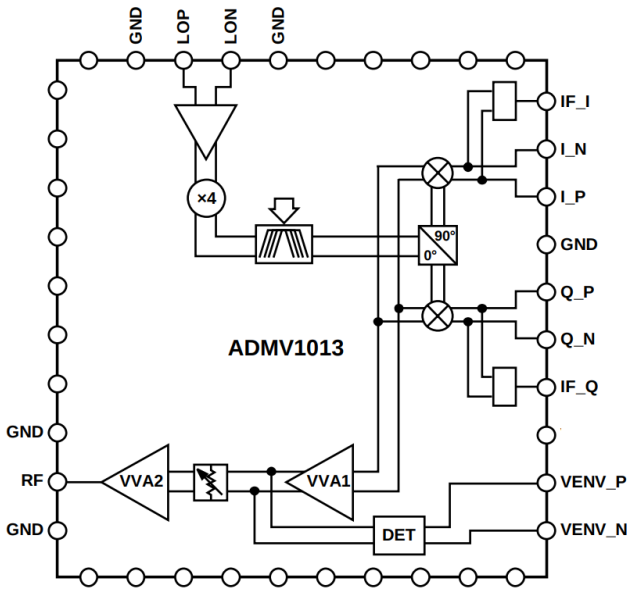


Figure 5.2: ADMV1013 internal block diagram [43].

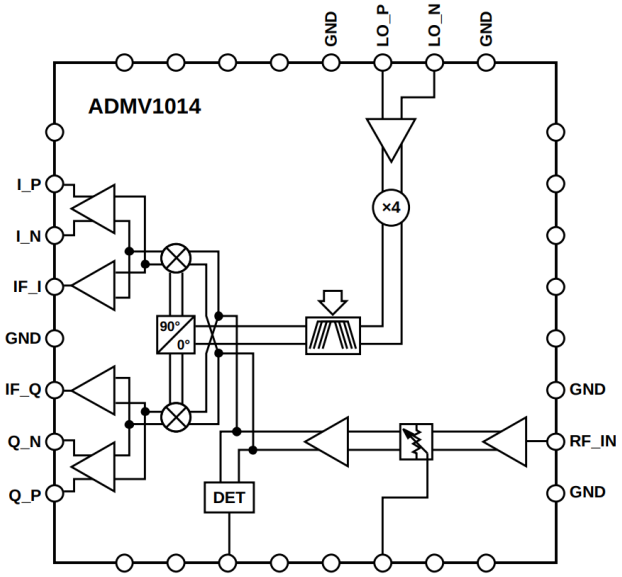


Figure 5.3: ADMV1014 internal block diagram [44].

To avoid additional losses, the RF input and output ports of the down and upconverter, respectively, are directly routed to 2.92 mm (K) connectors. The LO input of each converter IC is routed to the local oscillator distribution chain, as per the diagram of fig. 5.1, and the baseband connections are left floating, as explicitly recommended in the datasheet for operation in IF mode.

The IF connections of both integrated circuits is complex (IQ), thus requiring a hybrid coupler to be used in SSB conversion mode with a real (i.e., not complex) IF signal. As the converters seem to operate best with an IF frequency of 3 GHz, the QCN-34+ coupler from Mini-Circuits was chosen. This coupler presents at most a 4 degree phase unbalance and 1.2 dB amplitude unbalance in the entire band of operation, from 2500 MHz to 3400 MHz, which can be corrected with the internal converters' tuners accessible through their digital interfaces. In the transmitter side, a band-pass filter is placed before the coupler, while in the receiver side, the same filter is placed after the coupler. This filter attenuates out-of-band interferers that may be present in the IF signal. The filter used in both cases is the Mini-Circuits BFCV-2895+, which has a 3 dB passband from 2220 MHz to 3570 MHz, as can be seen by fig. 5.4. This fits perfectly with the used hybrid coupler, presenting a slightly wider passband. Both the filter and coupler share component footprints with components with different specifications, so that, even after the PCB is produced, some room for experimentation still exists. This also adds to the flexibility of the design.

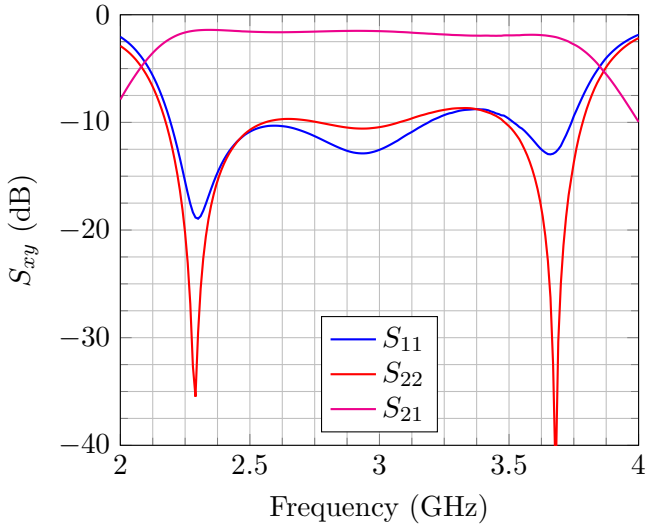


Figure 5.4: IF BPF S-Parameters.

5.2.2 *Internal LO generation*

The internal LO generation is based on the same synthesizer IC as used in the base platform, the 8V97003, from Renesas Electronics. It's characteristics and relevance for the use case have already been described in detail in section 3.2.3. The schematic for this block is the same as in the base platform, including the RF output network. As the synthesizer provides two independent RF outputs, one is routed to the transmission chain (the upconverter) and the other is used in the receiving chain (the downconverter).

5.2.3 *External LO conditioning*

The LO conditioning and distribution chain is presented in more detail at fig. 5.5. It must operate between 5.4 GHz and 10.25 GHz and offer a controllable gain to support a wide dynamic power range at the input. Contrary to the internal LO synthesizer, the external LO chain offers only a single input. Thus, a power divider is needed to feed both the transmitting and receiving chains. The following sections detail the approach in designing both the variable gain amplifier section and the power divider.

5.2.3.1 *Variable gain amplification*

The external LO conditioning comprises two amplifiers, a digitally controllable attenuator, and a power divider. This combination, as a whole, constitutes a variable gain amplifier, allowing different power levels at the input to be corrected on-board to produce roughly the same power level at the converters LO input.

The amplifiers are two PMA2-123LN5+ from Mini-Circuits, chosen mainly for their low noise figure of 1.4 dB in the band of interest, operating at an on-board available voltage of 5 V, self-biasing circuitry and low external component count requirements. Each amplifier presents a typical gain of 15.1 dB at 5 GHz and 11.6 dB at 10 GHz and an output 1 dB compression point of about 11.9 dBm in the band of interest (5.4 GHz to 10.25 GHz).

Between the two amplifier sits an ADRF5720, from Analog Devices. This 6-bit step attenuator has a configurable attenuation range of 0.5 dB to 31.5 dB in 0.5 dB steps, and works at up to 40 GHz, way more than required for this application. Both the attenuator and the amplifiers support operation without any external impedance

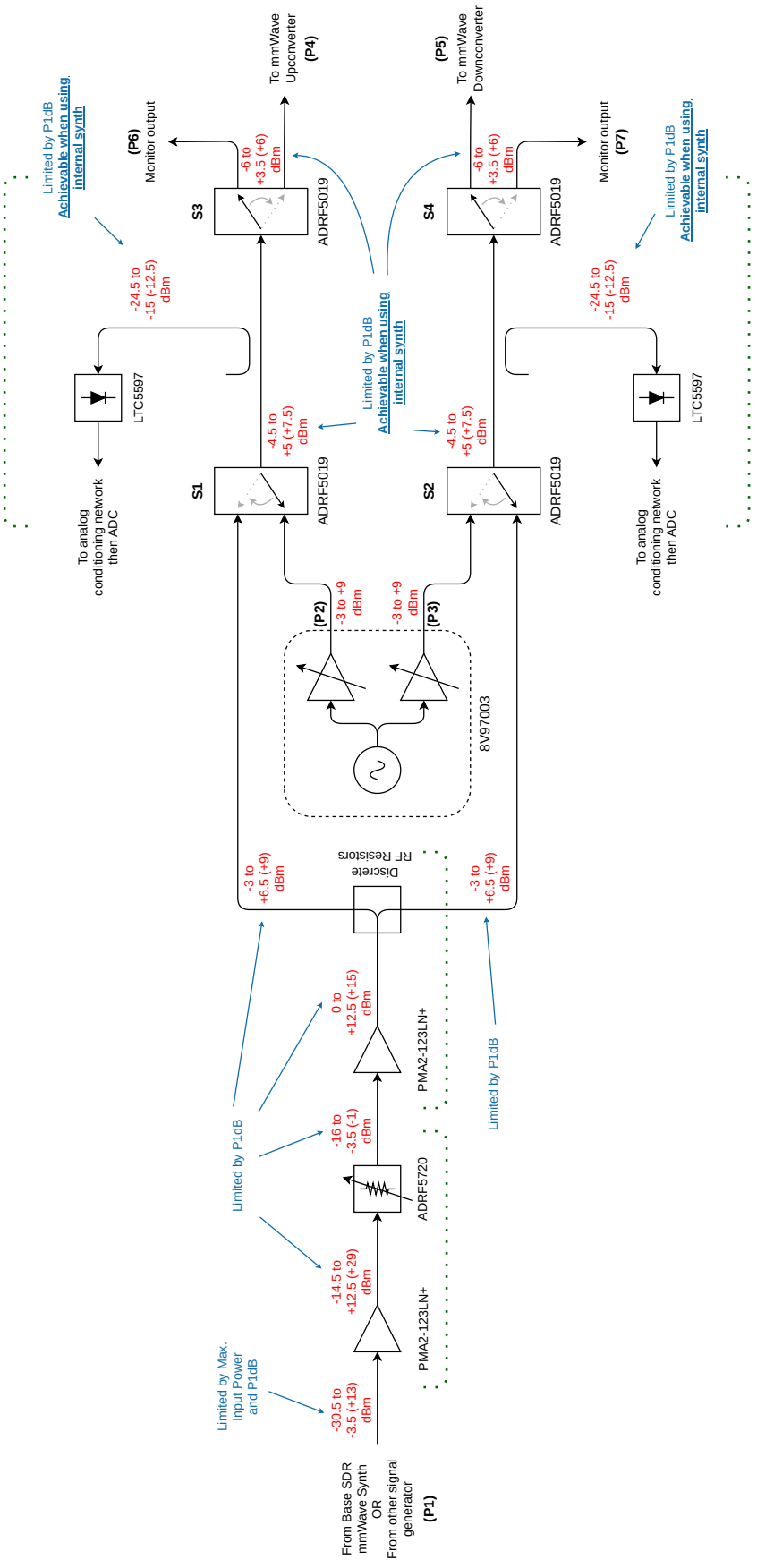


Figure 5.5: mmWave expansion card LO chain block diagram.

matching networks, due to the acceptable return loss present on their ports. However, full cascaded stage simulations are performed to evaluate the behavior of the entire chain of devices.

Figure 5.6 represents the physical layout of components and transmission lines of the external LO amplification block. It comprises two amplifiers with a variable attenuator in between, as previously described. The transmission lines are realized through microstrips sized to present a characteristic impedance (Z_0) of 50 Ω. The attenuator does not require any external component in the RF transmission lines, however, the amplifiers do.

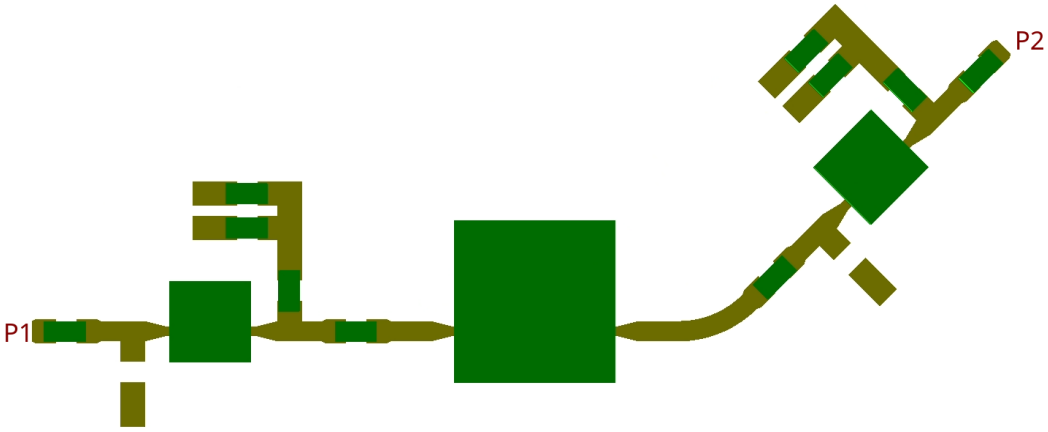


Figure 5.6: External LO variable gain amplification chain EM simulation layout.

The amplifier IC manufacturer specifies that no DC level should be present at the input of the amplifier, and, in the reference designs, suggests placing an inductor connecting the input to ground to enforce this condition. It does, however, not specify what exact component part number to use. A suitable component was selected from Coilcraft's 0402DC ceramic core inductor series, by analyzing their frequency response. It can be seen in fig. 5.7 that the 8.8 nH inductor behaves like an open circuit at the frequency of interest, while the other inductors do not. The reflection coefficient of the 1 nH and 3 nH inductors, both having one terminal shorted, are plotted in fig. 5.7 to demonstrate they would have a significant effect on the return loss of the amplifier shall they be used instead of the 8.8 nH part. An inductor with the same part number is used on the output side of the amplifier, to provide the necessary bias current, as recommended by the manufacturer.

To isolate the DC level present on the transmission lines, a capacitor is placed between blocks. This capacitor, from the ATC 600L family, is designed to operate at the RF frequencies involved in this design, and presents itself almost as a short circuit in the band of interest, as illustrated by the reflection coefficient of it in a

series connection on fig. 5.8. This is the desired behavior, as opposed to the inductor analyzed before, as this capacitor is placed in series with the different blocks and should not affect the impedance matching between them, while the inductor is placed in a shunt configuration.

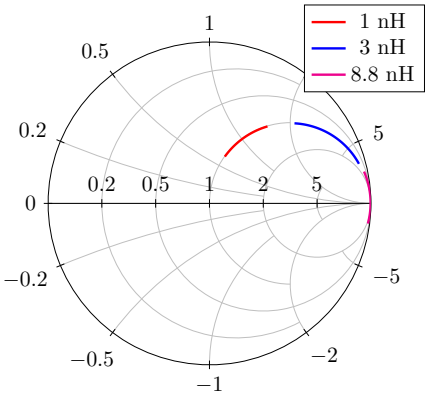


Figure 5.7: Return loss of three inductors of Coilcraft's 0402DC series, from 5 GHz to 11 GHz, with one of the terminals shorted to ground.

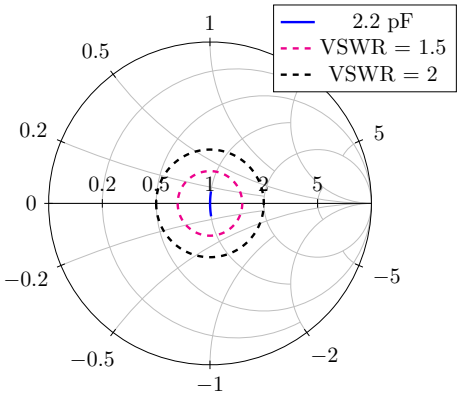


Figure 5.8: Return loss of a 2.2 pF ATC 600L microwave capacitor, from 5 GHz to 11 GHz, with one pad terminated with an ideal 50 Ω load.

Figure 5.9 traduces the electromagnetically simulated performance of the circuit, in the form of it's scattering (S) parameters. The simulation was performed with the variable attenuator at it's minimum attenuation setting. S_{21}, which represents the small-signal gain of the block (when terminated to Z_0), is above 27.5 dB in the whole band of interest, and above 30 dB up to 8.75 GHz. The output return loss, S_{22}, is below -20 dB up to 9.5 GHz, and below -15 dB on the entire operation frequency range. This is of paramount importance, as it can significantly affect the balance of the power divider, as will be seen next. The input return loss given by S_{11} is acceptable, presenting a value below -10 dB for most of the band, and never rising above -5 dB. The plot in fig. 5.10 shows the simulated stability assessment parameters: Rollett's stability factor, K, in pink, and $B1$, in blue, over a frequency range up to 20 GHz. These factors are used to evaluate the unconditional stability of the external LO amplification chain. As can be seen, $K > 1$ and $B1 > 0$ for the entire plotted frequency range, which ensures the chain is unconditionally stable. It is important to assess these parameters across a wider frequency band than the band of interest, since any instability, even outside the operation frequency band, could lead to unwanted oscillations, which can degrade the amplification and linearity of the external LO conditioning circuit, negatively impacting the external LO phase noise performance or even making it impossible to use the external LO distribution chain. The data in the plot is obtained with the variable attenuator at

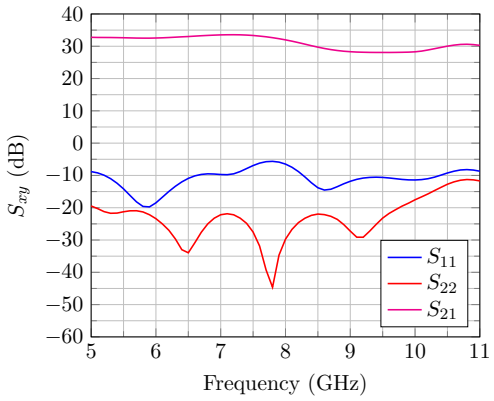


Figure 5.9: External LO variable gain amplification chain EM simulated S-Parameters.

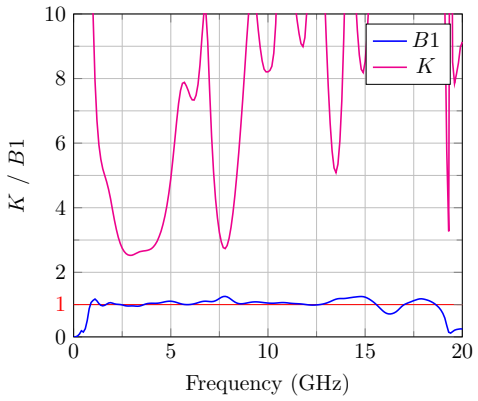


Figure 5.10: External LO variable gain amplification chain stability parameters.

the lowest attenuation setting, which represents the worst-case scenario. Figure 5.11 depicts the variation in the gain of the entire LO conditioning chain with different attenuator settings. As expected, the entire frequency range is attenuated by the same amount.

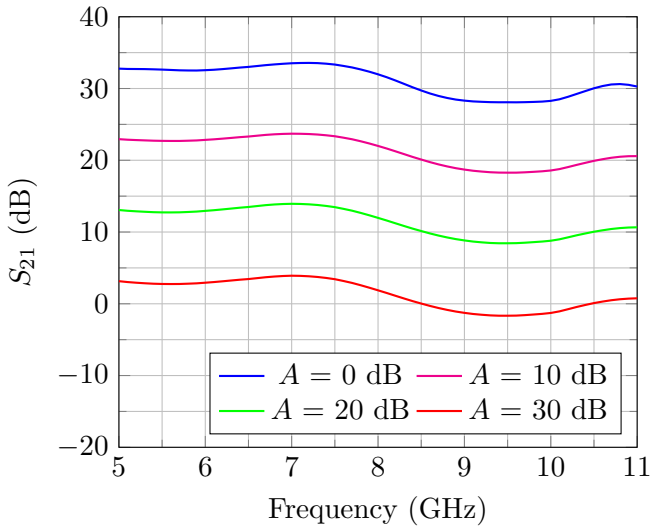


Figure 5.11: External LO variable gain amplification chain gain (S_{21}) variation with attenuator setting (A).

5.2.3.2 *Power divider*

Due to the wide operation band of the external LO signal, a reactive power divider implementation is quite challenging and space consuming. For this reason, a resistive power divider is instead designed using Vishay's thin film high-frequency resistors from the FC0402 series, interconnected with microstrip transmission lines.

The layout, visible in fig. 5.12, corresponding to the structure depicted in fig. 5.13, was then electromagnetically simulated and tuned to achieve the best performance in the required operation frequency band. The simulation results of the optimized structure are presented in fig. 5.14. Due to the symmetric behavior of the structure, the return loss of the three ports (S_{11}, S_{22} and S_{33}) is very similar, differing by less than 0.5 dB (due to the small implementation and component variations). The same applies to parameters that represent transmission loss (S_{21}, S_{31}, S_{32}, S_{12}, S_{13}, and S_{23}). To avoid cluttering the plot, only one representative value of each performance parameter is shown.

It can be observed that the return loss of lower than -16 dB indicates a very good 50 Ω impedance match across the entire presented frequency band. The transmission loss does not exceed 6.7 dB, and considering the minimum theoretical achievable

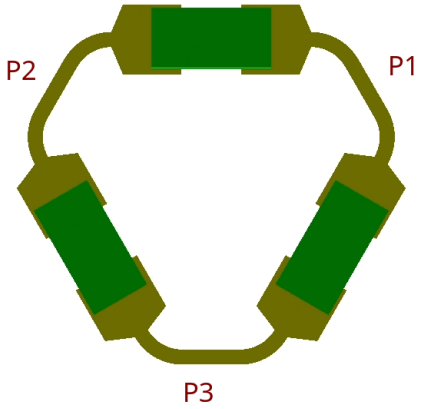


Figure 5.12: Power splitter EM simulation layout.

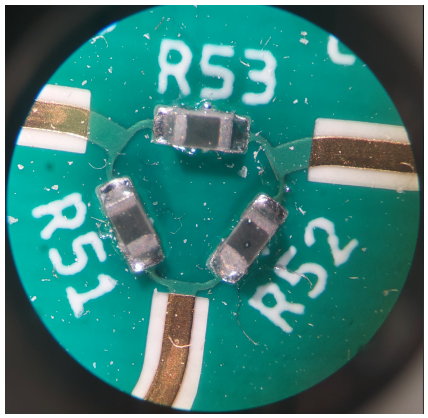


Figure 5.13: Power splitter layout on the PCB prototype.

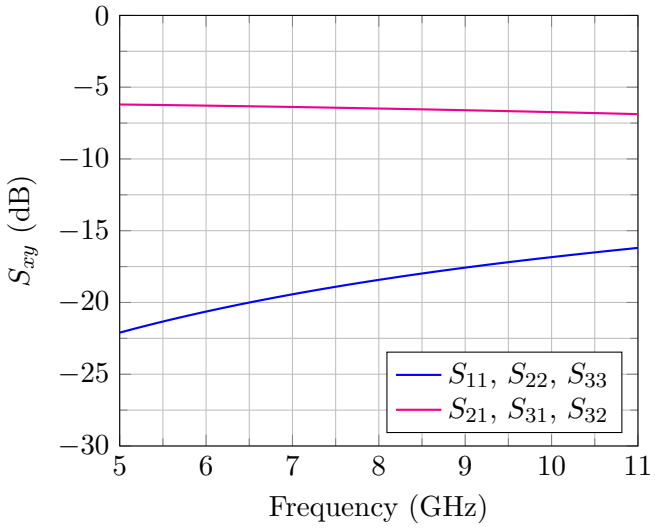


Figure 5.14: Power splitter EM simulated and optimized S-Parameters.

loss for this circuit topology is approximately 6.02 dB, the losses introduced by this circuit are under 1 dB.

5.2.4 *LO switching and monitoring*

As illustrated in fig. 5.5, the external and internal LO signals converge at two RF switches (S1 and S2), which allow selecting between the internal and external LO signals. From this point on, there are two similar and mirrored signal paths, one ending at the mmWave upconverter probe switch (S3) and the other one at the downconverter probe switch (S4). The four RF switches used in the design are all non-reflective, with part number ADRF5019, from Analog Devices. They operate from 100 MHz to 13 GHz, and have a typical insertion loss of 0.8 dB in the center of the design operation band. The non-reflective nature of the switch is crucial in this design, meaning the unused LO signal is properly terminated to the system impedance, minimizing unwanted reflections and, especially, maintaining the power balance of the resistive divider.

At the output of the first RF switch in each path (S1 and S2), there is a directional coupler to sample the signal. This coupler is designed using microstrip transmission lines, and, similarly to the resistive power divider, the layout is EM simulated and optimized to achieve the best performance in the band of interest. The isolated port of the coupler is terminated using an FC0402 RF resistor, and the coupled port is connected to an LTC5597 linear-in-dB RF power detector, from Analog Devices. The detector operates up to 70 GHz with a 35 dB dynamic range and an error of ±1 dB. The output voltage of the power detector is conditioned and monitored by the control system to obtain and regulate the LO signal power at any given time. The aforementioned structure layout, used for EM simulations, is depicted in fig. 5.15.

The electromagnetic simulation results, after optimization, are displayed in fig. 5.16. The insertion loss of the coupler, traduced by S_{21}, is better than 0.1

Figure 5.15: Directional coupler EM simulation layout.

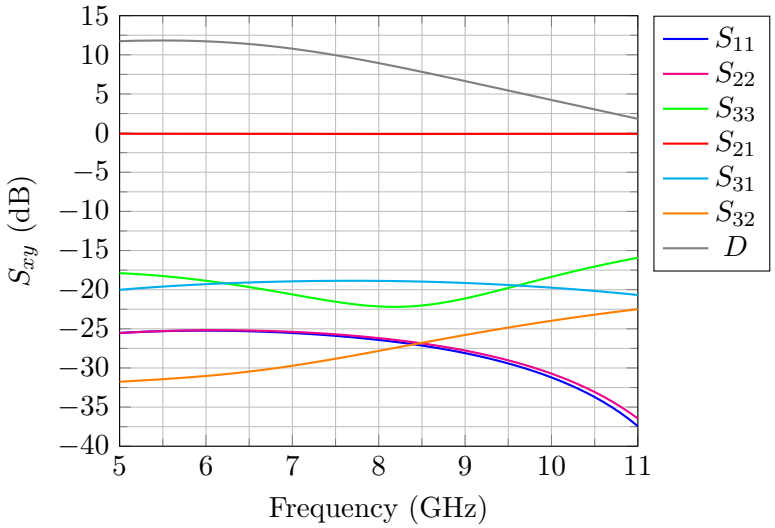


Figure 5.16: Directional coupler EM simulated S-Parameters.

dB, minimizing the impact of the signal traveling through the coupler to its destination. The impedance match of the input and output ports, S_{11} and S_{22}, respectively, is also very good, presenting a value better than -25 dB, while the impedance match of the probe (coupled) port (S_{33}) is better than -17.5 dB. The target coupling factor (S_{31}) of this design is -20 dB, and its layout was optimized to minimize variation of this parameter over the 5 GHz to 10 GHz band. The achieved coupling factor is within 0.3 dB of the target value at the band edges, while reaching a slightly higher value of -18.9 dB in the center of the band. The isolation (S_{32}) is better than -23 dB in the whole band of interest, while the achieved directivity, D, given by $S_{31} - S_{32}$, is better than 10 dB in the lower half of the operation band, while falling slightly below 5 dB near the upper end of the band (10 GHz). Although the directivity is relatively low at the higher end of the frequency band of interest, it does not have significant practical consequences due to the high return loss (greater than 20 dB up to 10 GHz) of the non-reflective switches (S3 and S4) used (ADRF5019).

After power monitoring, the signal enters the common terminal of another RF switch (S3 and S4, one for each LO path), which offers the ability to route the signal directly to the up/downconverter, during normal operation, or to an external connector for outside monitoring and calibration.

5.2.5 *Control and Monitoring*

The control and monitoring section of the mmWave expansion card is based on an ARM Cortex-M4F from Silicon Labs' EFM32 Giant Gecko series. The Cortex-M4F

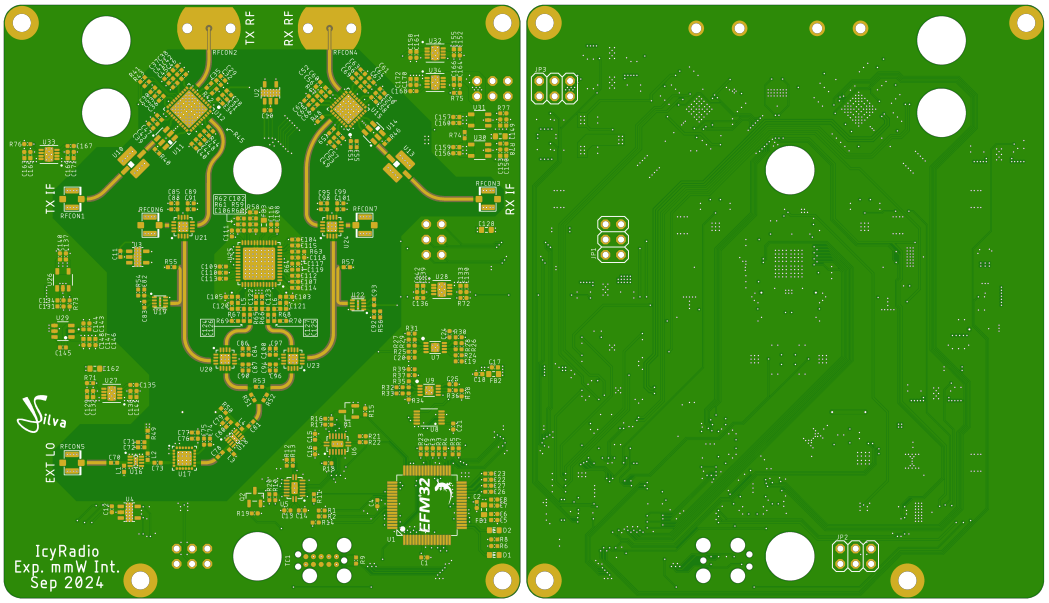
offers quite a lot of computing power, including a floating-point unit (FPU), needed to implement the power control loops discussed previously, to quickly calculate the coefficients to write to the internal synthesizer to set the frequency, among other tasks. Silicon Labs also pack the device with rich analog features, aiding in the accurate measurement of analog voltages generated throughout the system, crucial for control. This MCU is the interface between all the on-board logic and the base platform, as it communicates with the SDR FPGA via I2C.

5.3 MMWAVE BOARD ESTIMATED PERFORMANCE

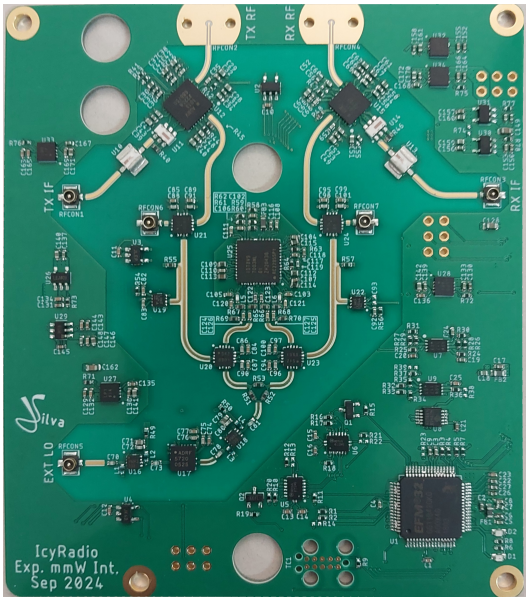
This section presents the estimated performance of the mmWave expansion card, focusing on key aspects of its design and evaluation. It is divided into two subsections: the first discusses the design considerations and layout choices, based on the designs outlined in section 5.2. The second subsection provides a detailed analysis of the board's simulated performance, evaluating key metrics such as signal integrity, loss, and overall functionality.

5.3.1 *PCB layout*

The top layer of the PCB is used mainly for routing the RF traces, all as microstrip lines, and for short traces connecting signals in the lower layers to surface mount component pads. The second layer (counting from the top) is entirely dedicated to ground, no exceptions are made. This ensures a large uninterrupted return path for all the current flowing through the PCB signals. The third layer is dedicated to power nets. Power is routed using big pours which span the entire area of the PCB where that specific power rail needs to connect, and is brought to the top layer using vias. The bottom layer is used for miscellaneous signal routing. It is also used, in very few cases where two power planes cross, to route power nets. All the empty space on all layers is filled with ground pours, except on the top layer, near microstrip lines. Figure 5.17 shows both a render of the finished PCB and a picture of the fully assembled prototype PCB. In fig. 5.17a, it is possible to see where the ground plane has been retracted so it does not interfere with the microstrip transmission lines. The mmWave up and downconverters are located at the top center of the PCB, while the synthesizer is located at the center, and the external LO conditioning chain is just below, a little bit towards the left. The voltage regulators are scattered around the RF circuit, so that they sit close to the



(a) (b)



(c)

Figure 5.17: mmWave expansion card PCB layout: (a) rendered top layer; (b) rendered bottom layer; (c) assembled prototype.

block they are powering, as mentioned previously. The MCU sits in the bottom right of the PCB. Just above the MCU sits the analog signal conditioning of the control and monitoring system, and to the left of the MCU sits the digital interface to the up and downconverters. In fig. 5.17b some traces can be seen, as well as the footprint for the connectors which interface with the base SDR platform.

5.3.2 *Performance assessment*

Before fabrication, the entire chain of devices, as laid out on the PCB, has also been simulated electromagnetically in order to assess potential issues that may not be visible through the individual block simulation results. The layout used in this simulation is depicted in fig. 5.18, and comprises the entire LO conditioning and distribution chain. As the integrated up and downconverters rely on non-linear properties to perform their function and because there are no models that capture their performance that can be used, they have been omitted from this simulation for simplicity. This is also possible due to the short length of the transmission lines employed in those sections (up and downconverter) of the circuit.

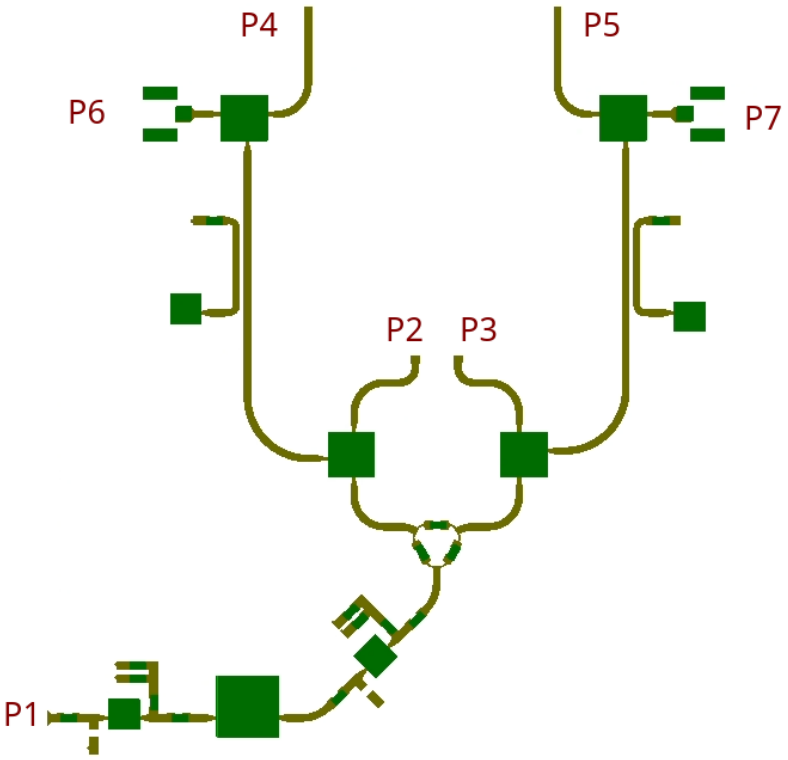


Figure 5.18: Complete LO conditioning and distribution chain EM simulation layout.

Due to the extended number of ports in this model, the most relevant results are presented in a comprehensive manner in figs. 5.19 and 5.20. In fig. 5.19, it is possible to observe the S-Parameters of higher importance that directly represent the performance of the circuit when operating the mmWave downconverter with an external LO, through correct positioning of the relevant RF switches. S_{51}, which is always above 20 dB in the band of interest, represents the gain applied to the external LO input (P1) before reaching the downconverter (P5). Most of the difference between this parameter and the results from fig. 5.9 comes from the loss

in the resistive power divider and the RF switches, as expected. The input return loss, S_{11}, follows the results from from fig. 5.9, since there is no additional circuitry with relevant effect here, while the output return loss, S_{55}, is kept below -10 dB for the majority of the operation band, slightly peaking above -10 dB near 9 GHz. Another parameter of paramount importance is the amount of leakage from the internal LO signal that appears at the downconverter when the external LO is selected. This figure is related to the S_{53} parameter, which is always better than -33 dB in the entire band of interest.

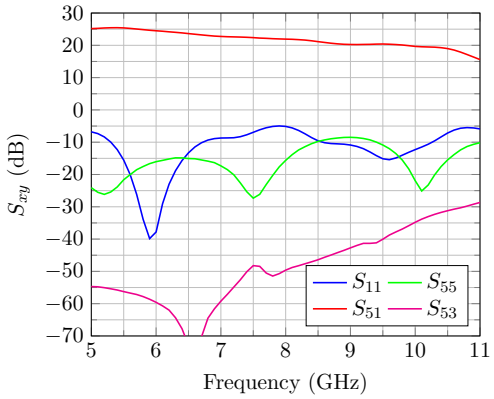


Figure 5.19: Complete LO signal chain simulation results, with external LO routed to the downconverter.

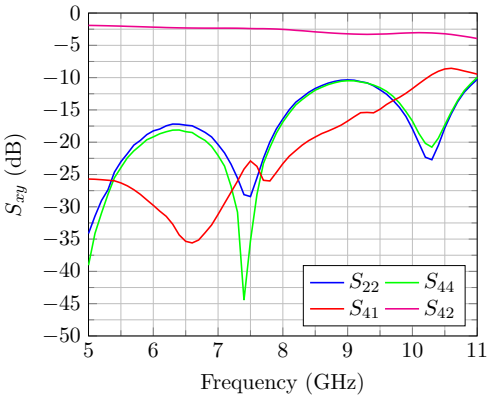


Figure 5.20: Complete LO signal chain simulation results, with internal LO routed to the upconverter.

Figure 5.20, in contrast, represents the opposite situation, i.e., the performance of the circuit when delivering the internal LO signal to the upconverter. This is a purely passive path, and, as such, the gain, S_{42}, is negative. It is, however, better than -6 dB between 5 GHz and 10 GHz, which should ensure enough power transfer between the synthesizer and the upconverter (and/or the downconverter). The input and output return loss curves, S_{22} and S_{44}, respectively, are very similar in behavior. This is again expected for the selected LO path. They, for the most part, present a good impedance match, with a value better than -15 dB. Near 9 GHz, the match is slightly worse, but sill acceptable nonetheless, never exceeding -10 dB. The S_{41} parameter related to the power of the signal appearing at the upconverter that originated at the external LO input port, while the internal LO path is selected. This is a relevant parameter when calculating isolation between both LO sources. Despite the apparent low isolation which may be indicated by the value of this parameter at 10 GHz (-10 dB), it is worth noting that there are other important settings when determining isolation.

The results presented in both figures are obtained with the variable attenuator at its lowest attenuation setting, and this may not always be the case. Depending on the external LO power level, the attenuation can be raised, and thus S_{41} will decrease, naturally. Isolation between both LO signals is a highly dynamic parameter which can be digitally maximized via the power monitoring and control system which is implemented in the mmWave expansion card.

5.4 FIRMWARE AND SOFTWARE

This section provides information about the algorithms that run in the ARM Cortex-M4F micro-controller unit of the mmWave expansion card, which mainly cover the communication with the base SDR platform, control algorithms for on-board power leveling loops, interfacing with the LO synthesizer and the mmWave signal converters.

The software architecture of this microcontroller is very similar to the base platform's PMU, addressed in detail in section 4.4.1. As the processor core architecture is the same, many sections of the software framework and environment can be reused, mainly the MCU peripherals and communication interfaces initialization process, which happens on boot.

The slave I2C interface algorithms and logic is also reused, as this microcontroller also presents itself as a slave on the I2C bus which connects directly to the base platform through the connectors described in section 3.2.6. The MCU in the mmWave expansion card never behaves as a bus master on that I2C bus, leaving the FPGA as the single master. A secondary I2C bus of the MCU is configured as master to independently interface with PCB temperature sensors without disrupting traffic between the FPGA and the expansion card.

A key aspect of the interface between the base platform and the expansion card is the automatic enumeration and discovery capabilities. Due to the usage of the I2C bus, it is possible for the FPGA to perform a bus scan to detect all the slaves present on the bus (i.e., expansion cards). The non-volatile memory of the MCU is used to store operational parameters of the expansion card, such as working frequency bands, power levels and calibration data. This data is then read by the base SDR, via the I2C interface, which automatically incorporates the capabilities of the expansion card in the list of attributes presented to the user through the SoapySDR API. Applied to all future expansion cards, this feature allows seamless integration and modularity of the entire system.

The LO power control loop is based on the data acquired from the power sensors, through the MCU ADC. A multi-point calibration is performed which allows accurate correlation between the ADC sampled voltage and the LO power level. This information is then used to regulate the power of the relevant (i.e., external or internal, according to the RF switches) signal. The external LO signal power is controlled through the variable attenuator, which is interfaced via SPI, and the internal LO power is regulator via the configurable synthesizer's output stage, which offers twelve discrete levels of output power.

6

SYSTEM PERFORMANCE EVALUATION

This chapter presents the practical performance evaluation of the developed systems using laboratory instruments for measurement and analysis. The first section assesses the base SDR platform, evaluating its performance. The second section focuses on the mmWave expansion card, performing bench tests with specialized equipment to characterize its capabilities and performance metrics. Finally, the chapter concludes with a practical demonstration, where the complete SDR system is used to transmit and receive a live video feed, validating the combined functionality of the base SDR platform and the mmWave expansion card in a real-world use case scenario.

6.1 SDR BASE PLATFORM

To evaluate the base SDR platform, the transmitter outputs were connected to an Agilent E4448A spectrum analyzer. As referred in section 3.2.2, one of the RF transceiver outputs, TX1A, is exposed directly to the user, unlike TX1B, which is first routed through an ADL5601 amplifier. Bench evaluation was performed on both outputs, in order to evaluate the different performance of the raw and amplified outputs. Figure 6.1 highlights the connection diagram depicted in the bench setup of fig. 6.2, which was used to obtain the results shown in table 6.1 and figs. 6.3 and 6.4. With the SDR configured to output a single tone at various RF frequencies with the maximum power setting, the output power of the tone was measured using the spectrum analyzer, and recorded in table 6.1. It is possible to

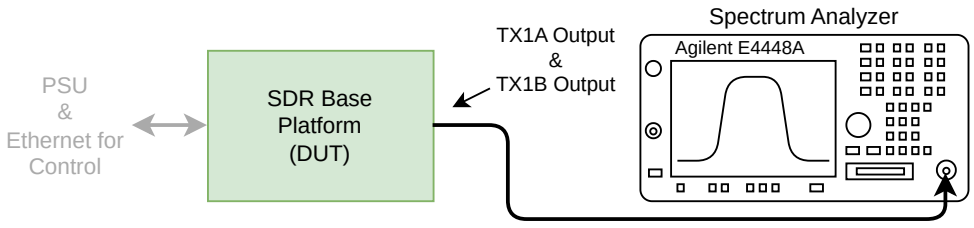


Figure 6.1: Instruments and connection diagram of the bench setup for testing the base SDR platform transmitter.

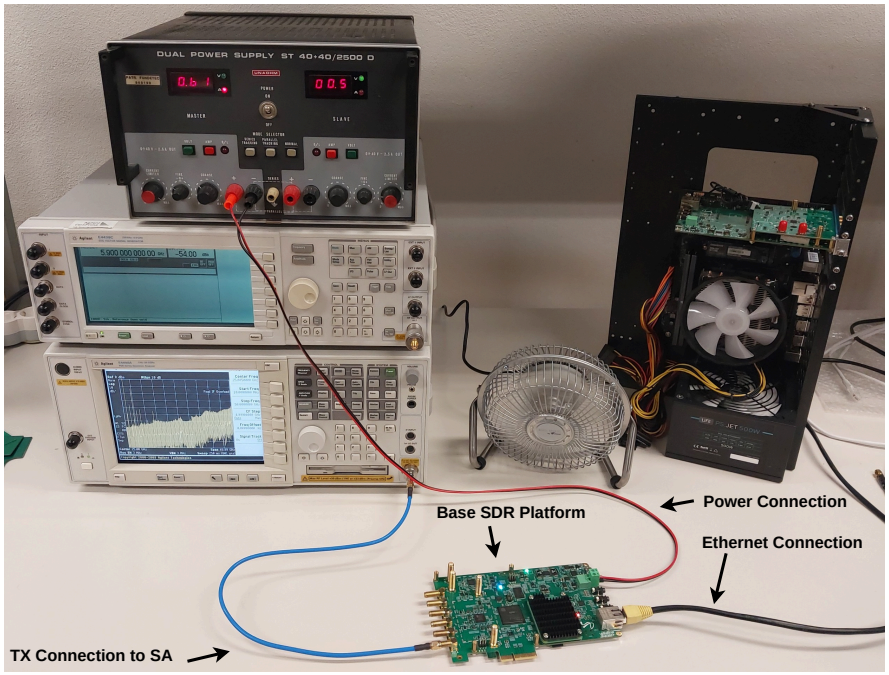


Figure 6.2: Bench setup for testing the base SDR platform transmitter.

observe the difference in output power between the TX1A and the TX1B output, as expected, the TX1B output provides a higher power output. For each of the six measured conditions listed in table 6.1, the output signal spectrum is depicted in fig. 6.3, up to 50 GHz. At 1 GHz (figs. 6.3a and 6.3b), it is possible to observe that the spurious tone with higher power is the third harmonic, at 3 GHz, and it is more than 15 dB below the fundamental tone. In the center of the operating band, at 3 GHz (figs. 6.3c and 6.3d), the second harmonic is the only visible spurious tone, and is more than 40 dB below the fundamental tone, while at 6 GHz (figs. 6.3e and 6.3f) all the harmonic content is below the noise floor of the instrument, given its configuration parameters (e.g., resolution bandwidth, input attenuation, etc). As with most wideband RF devices, the application system where the SDR will be employed must provide output filtering to avoid disturbing neighboring channels.

Table 6.1: Summary of the fundamental tone output power of the SDR base platform.

Frequency	**TX1A[1] Fund. Power**	**TX1B[2] Fund. Power**
1 GHz	-8.38 dBm	6.74 dBm
3 GHz	-13.19 dBm	1.00 dBm
6 GHz	-19.10 dBm	-10.49 dBm

[1] RF output directly connected to the transceiver.
[2] RF output from the ADL5601 gain block

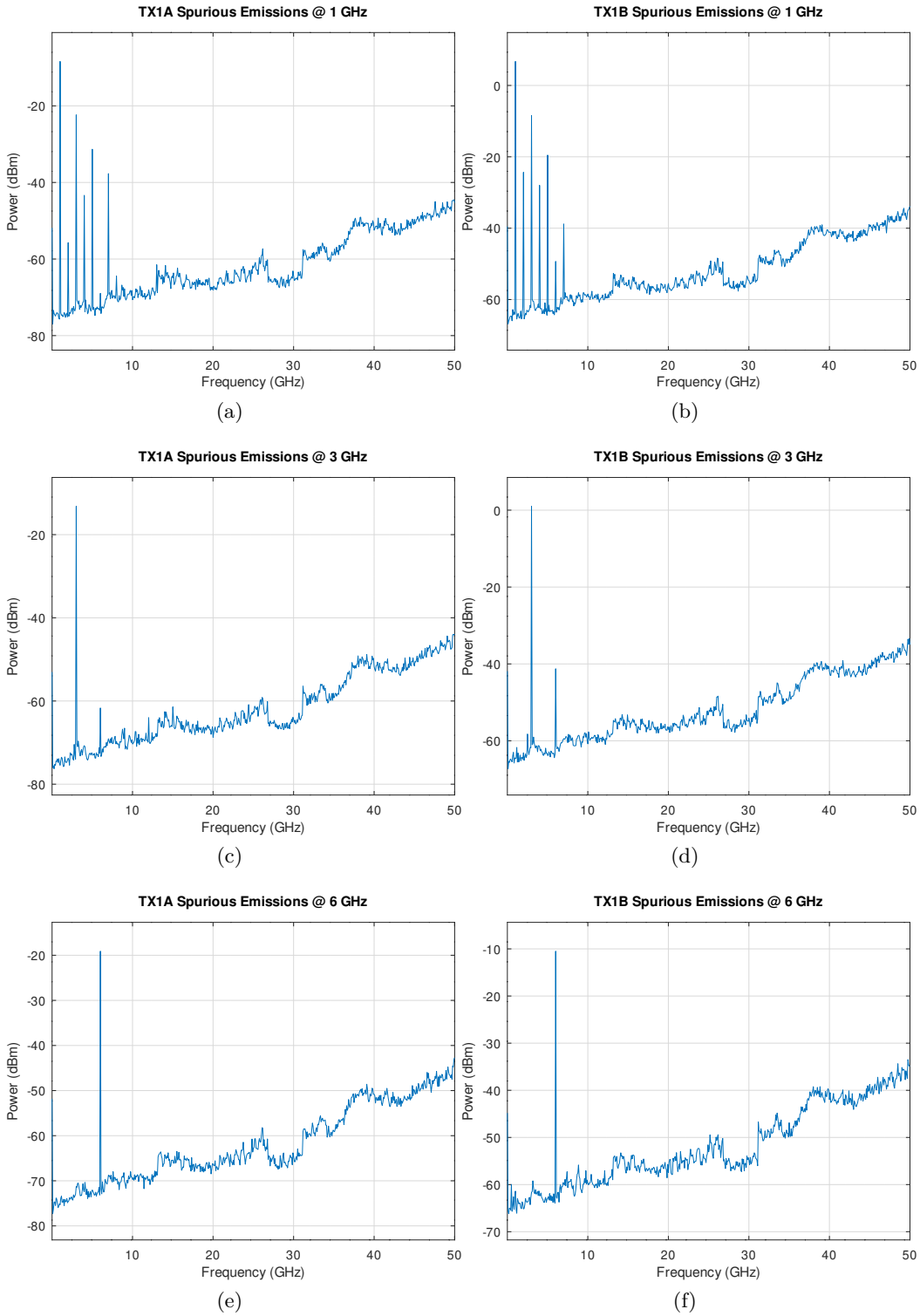


Figure 6.3: Base SDR platform output spectrum while transmitting a single tone at maximum power: (a) f_{RF} = 1 GHz, directly output from the RF transceiver; (b) f_{RF} = 1 GHz, output via the on-board gain block; (c) f_{RF} = 3 GHz, directly output from the RF transceiver; (d) f_{RF} = 3 GHz, output via the on-board gain block; (e) f_{RF} = 6 GHz, directly output from the RF transceiver; (f) f_{RF} = 6 GHz, output via the on-board gain block.

Following the single tone test, the SDR was configured to output a 64-QAM modulated RF carrier centered at 3 GHz. The modulating signal was shaped by a root-raised cosine (RRC) filter with roll-off coefficient, α, of 0.2, achieving an occupied bandwidth of 2 MHz. Two attenuation levels were tested on each RF output, totalling four different combinations, as depicted in fig. 6.4. Table 6.2 summarizes the measured adjacent channel power ratio (ACPR) of each combination, which is always better than -35 dBc.

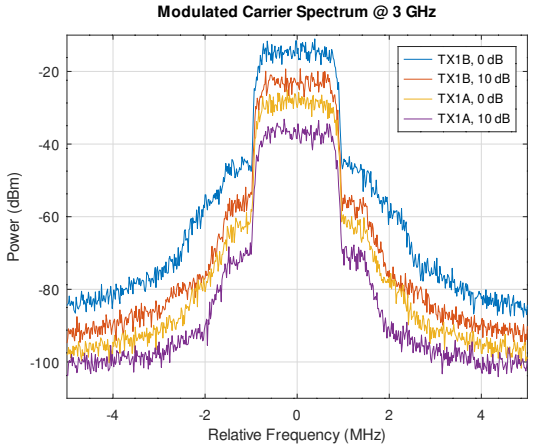


Figure 6.4: Base SDR platform output spectrum of a 64-QAM modulated carrier at $f_{RF} = 3$ GHz, output directly from the RF transceiver (TX1A) and via the amplified output (TX1B), at maximum power and with $A = 10$ dB.

Table 6.2: Summary of the ACPR of a 64-QAM modulated carrier at $f_{RF} = 3$ GHz output from the SDR base platform.

Attenuation	**TX1A**[1]	**TX1B**[2]
0 dB	-38.17 dBc	-35.81 dBc
10 dB	-38.63 dBc	-38.46 dBc

[1] RF output directly connected to the transceiver.
[2] RF output from the ADL5601 gain block

With the results just presented, it is possible to consider the SDR transmitter performance to be good. Nonetheless, there are some parameters that require improvements, mainly, the output power of the SDR does not meet the initially set goal of +10 dBm up to 6 GHz. This limitation has been investigated, and the cause has already been identified to be related to the output connectors on the board. More information is available further in this thesis, in section 7.2.

6.2 MMWAVE EXPANSION CARD

To evaluate the achieved performance of the mmWave expansion card's LO amplification and distribution chain, a Copper Mountain Cobalt Series C4420 4-port vector network analyzer (VNA) has been utilized, as can be seen in the connection diagram of fig. 6.5. Figures 6.6 and 6.7 illustrate the bench setup utilized in in the next measurements of the external LO conditioning and distribution chain. The instrument was connected to the expansion card's external LO input (P1 in fig. 5.18), as well as the TX and RX LO monitoring port (P6 and P7, respectively, in fig. 5.18). The LO monitoring switches (S3 and S4 in fig. 5.5) were configured to route the LO signals to the probe connectors instead of the up and downconverter ICs, while S1 and S2 were configured to select the external LO signal path.

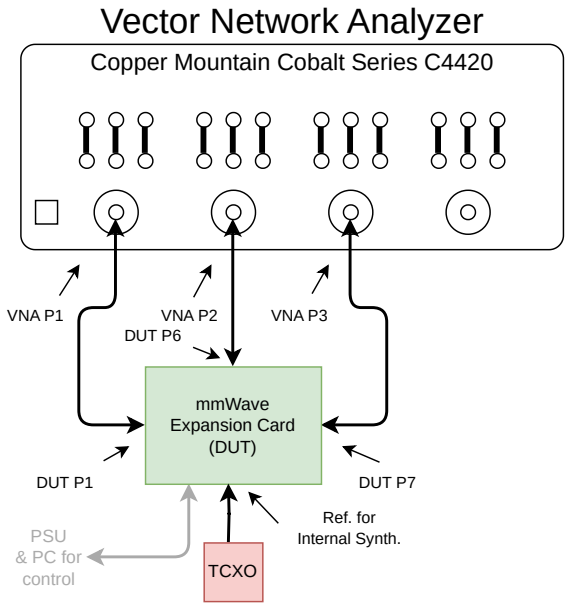


Figure 6.5: Instruments and connection diagram of the bench setup for testing the mmWave expansion card LO distribution chain.

With this bench setup, the external LO conditioning chain gain was measured, obtaining the results presented in fig. 6.8. This figure depicts the measured small-signal gain of the entire chain (S_{61}) at four different attenuation levels, as set by the on-board variable attenuator. Also overlayed in the plot of fig. 6.8 is the simulation results of this parameter. It is possible to observe that the measured value closely follows the simulated one, although the measured value presents some ripple. This ripple may be caused by the u.FL pigtails used in the measurement setup, which allow the connection of the VNA to the mmWave expansion card, as shown in fig. 6.7.

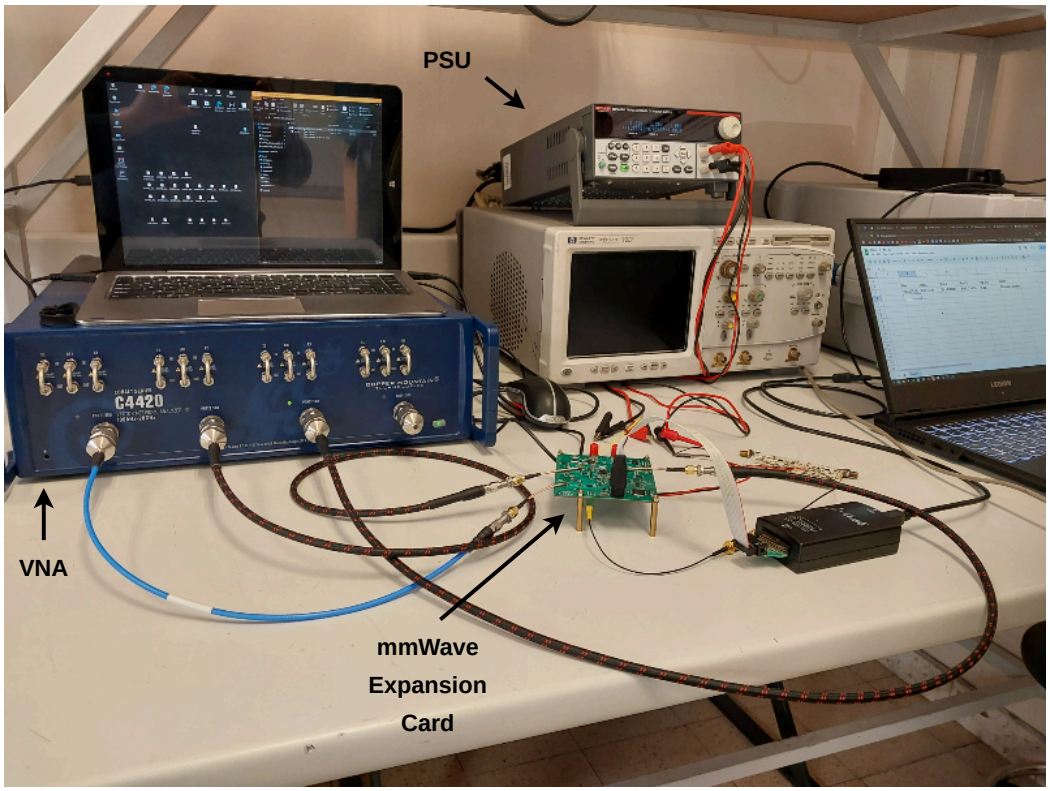


Figure 6.6: Bench setup for testing the mmWave expansion card LO distribution chain.

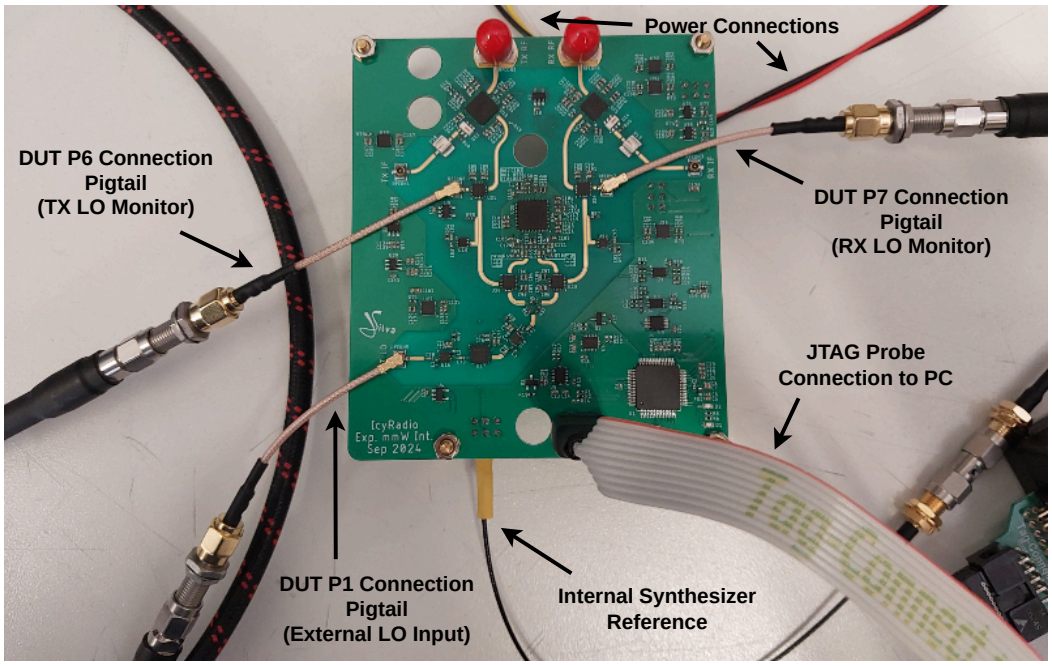


Figure 6.7: Connections of the Copper Mountain VNA ports to the mmWave expansion card for testing the LO distribution chain.

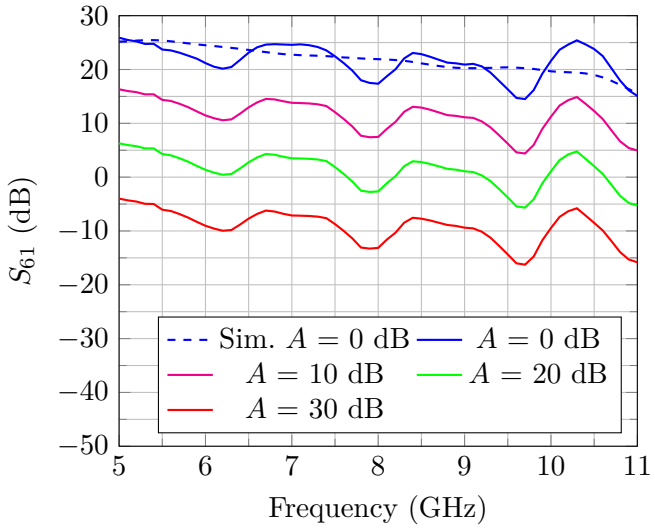


Figure 6.8: Measured mmWave expansion card external LO gain (S_{61}) variation with the variable attenuator setting (A).

Using the same setup of fig. 6.5, the RX LO selection switch (S2 in fig. 5.5) was toggled, routing the internal LO signal to the upconverter, while the TX LO selection switch (S1) was left in the external LO position. After this, S_{61} was again measured with the VNA, showing no differences when compared to the initially measured value. This measurement allowed the assessment of the RF switches' internal termination load, as if a poor 50 Ω impedance match was present, the balance of the RF power divider would have been disturbed, and, thus, S_{61} would change.

To assess the mmWave up and downconverter performance, a bench was first prepared with an Agilent ESG E4438C signal generator and an Agilent E4448A spectrum analyzer. As illustrated in fig. 6.9, this setup was used in three different phases. First, to obtain reference values, such as signal power and distortion levels, the signal generator was directly connected to the spectrum analyzer, as represented by step 1 of fig. 6.9. The reference measurement allowed the correction of the results obtained in the following steps, by considering, for example, the attenuation introduced by the cables involved in the connections. Step 2 (of fig. 6.9) was used on the bench to assess the mmWave upconverter's conversion gain, spurious emissions, LO feedthrough, image rejection and distortion, using both a two-tone signal, to measure IP3, and a modulated carrier, to measure ACPR. The bench setup corresponding to step 2 is depicted in fig. 6.10, which was initially used to obtain the conversion gain of the mmWave transmitter, using a single tone signal from the signal generator as its IF input and measuring the power of the upconverter signal in the mmWave band with the spectrum analyzer. The results of this measurement

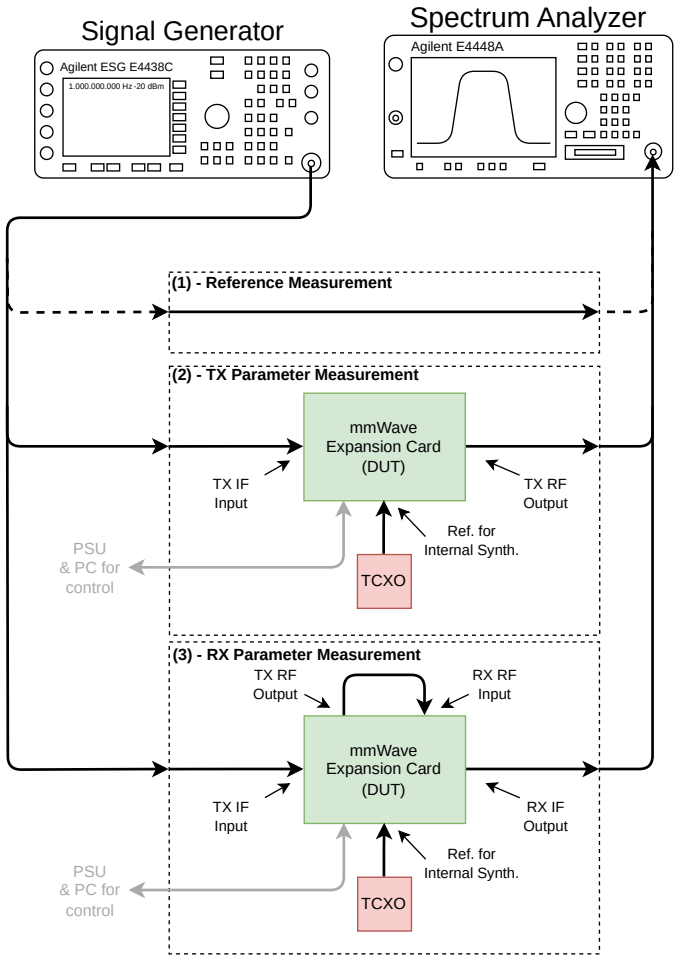


Figure 6.9: Instruments and connection diagram of the bench setup for testing the mmWave expansion card with a signal generator and spectrum analyzer.

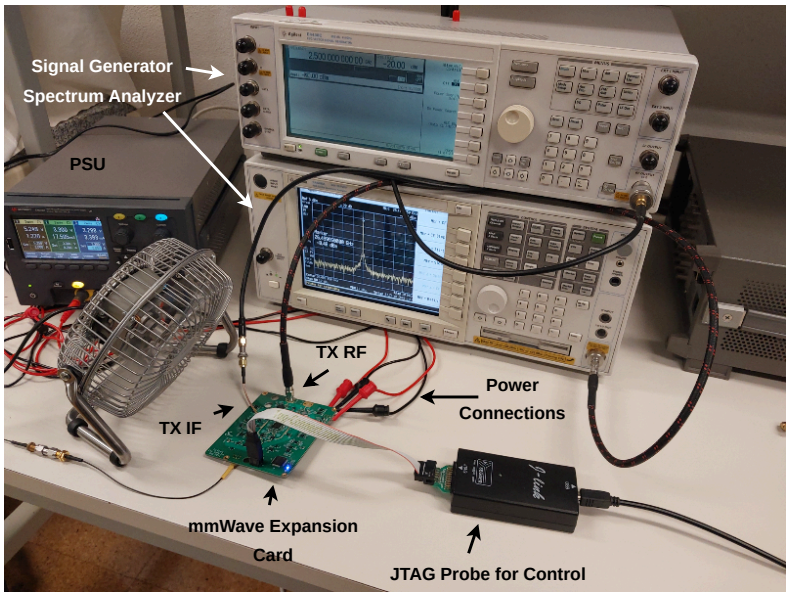


Figure 6.10: Bench setup for testing the mmWave expansion card with a signal generator and spectrum analyzer.

can be observed in table 6.3 for various combinations of RF and IF frequencies. As expected, the gain of the transmitter is higher at lower RF frequencies, achieving a minimum gain of 8.1 dB at the maximum RF frequency of 40 GHz. The gain variation with IF frequency is always better than 2.5 dB across all the measured RF frequencies, and, for the measured RF frequencies below 33 GHz, it is better than 2 dB. It can also be observed that the maximum gain at a specific RF frequency is always achieved when using an IF frequency of 2.5 GHz. It is worth noting that the presented values correspond to the the gain of the entire transmitter chain, as they include the insertion loss of the IF BPF and the hybrid coupler, as well as residual insertion loss of the connecting PCB traces. The loss introduced by the cables connecting the bench instruments to the mmWave expansion card were de-embedded from the measurements, as explained before.

Table 6.3: Summary of the mmWave expansion card transmitter conversion gain at different RF and IF frequencies and maximum gain ($V_{CTRL1} = V_{CTRL2} = 1.8$ V, control pins of the ADMV1013).

RF Frequency	**IF Frequency**	**Gain**
23 GHz	2.5 GHz	16.4 dB
	3 GHz	15.6 dB
	3.5 GHz	14.4 dB
26.5 GHz	2.5 GHz	15.9 dB
	3 GHz	15.0 dB
	3.5 GHz	14.3 dB
33 GHz	2.5 GHz	11.5 dB
	3 GHz	9.4 dB
	3.5 GHz	10.1 dB
40 GHz	2.5 GHz	10.3 dB
	3 GHz	8.1 dB
	3.5 GHz	8.8 dB

Maintaining the same input signal (a single tone), the spectrum analyzer was used to assess the spurious content of the upconverter's output. Figure 6.11 represents the results obtained during this test, depicting both full and narrow-span spectrum plots at 26.5 GHz and 40 GHz. In both figs. 6.11a and 6.11b, several tones can be seen. In these figures, the red traces correspond to measurements without an IF signal (i.e., IF port terminated to 50 Ω) and the blue ones are obtained with an IF signal present at the input of the upconverter. As described in section 5.2.1, the

upconverter (and the downconverter) receive a local oscillator signal at one fourth of the final RF frequency, multiplying it internally. The multiplication process, however, inherently produces several additional harmonics of the input LO tone. Each of these harmonics will be mixed with the IF signal, producing copies of this signal at each LO harmonic. Despite the IC including a filter to mitigate the side-effects of this unwanted behavior, it is still possible to see some unwanted tones in figs. 6.11a and 6.11b. In the red curves of both figures, only the LO harmonics can be observed, as expected. However, in the blue curves, the harmonic mixing behavior just described can be observed.

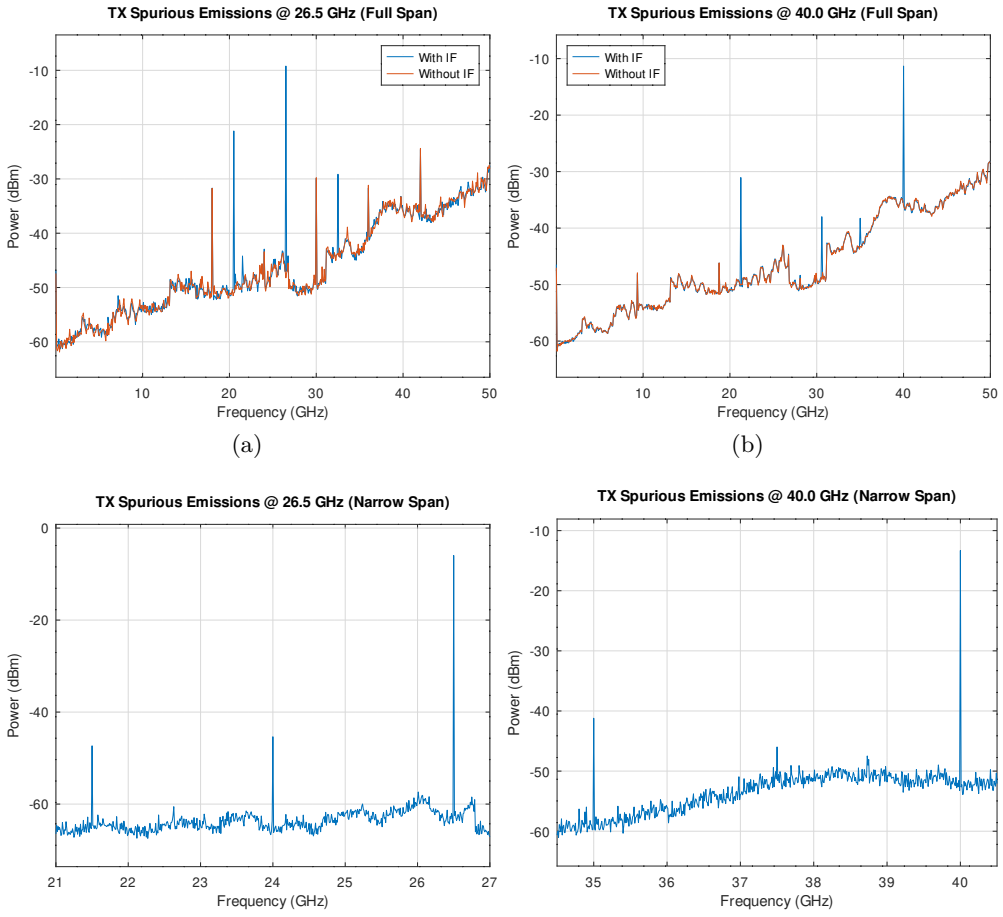


Figure 6.11: mmWave upconverter output spectrum: (a) Full span with f_{RF} = 26.5 GHz, with and without IF signal; (b) Full span with f_{RF} = 40 GHz, with and without IF signal; (c) Narrow span with f_{RF} = 26.5 GHz; (d) Narrow span with f_{RF} = 40 GHz.

As noted in section 5.2.1, both converter ICs contain digital tunable elements to balance the IF in-phase and quadrature components' phase and amplitude. By tuning those elements while observing the output signal spectrum of the upconverter,

it is possible to minimize the LO feedthrough and maximize the sideband (image) rejection. Figures 6.11c and 6.11d depict the narrow-span capture of the spectrum analyzer after performing the tuning process, while the image and LO rejection figures are summarized in table 6.4. It can be seen that, at 26.5 GHz, both the LO feedthrough and image rejection can be optimized to better than 40 dB, since these effects are mainly caused by I/Q unbalance. At 40 GHz, adjusting I/Q balance allows the transmitter to achieve acceptable levels of LO feedthrough and image rejection of 31.5 dB and 27.5 dB, respectively. At such a high frequency, these unwanted effects can be caused by other effects other than I/Q unbalance, such as coupling between signal paths via the IC's substrate.

Table 6.4: Summary of the mmWave expansion card transmitter LO and image rejection at $f_{IF} = 2.5$ GHz, different RF frequencies and maximum gain ($V_{CTRL1} = V_{CTRL2} = 1.8$ V).

Frequency	**LO Rejection[1]**	**Image Rejection[1]**
26.5 GHz	40 dB	42.49 dB
40 GHz	31.5 dB	27.5 dB

[1] Digital tuners adjusted for maximum LO and image rejection.

While still keeping the bench setup of fig. 6.9, the signal generator was configured to output a two-tone signal, followed by a 16-QAM modulated carrier with a bandwidth of 10 MHz, in order to assess the distortion performance of the mmWave transmitter. The OIP3 of the upconverter IC, assessed with two tones spaced 1 MHz apart from each other, at different RF frequencies, is summarized in table 6.5. The best performance is observed at 26.5 GHz, achieving a 17.25 dBm OIP3, while at other operating frequencies, the OIP3 is always higher than 13 dBm.

Table 6.5: Summary of the OIP3 of the mmWave expansion card transmitter output at maximum gain ($V_{CTRL1} = V_{CTRL2} = 1.8$ V).

Frequency	**OIP3**
23 GHz	15.5 dBm
26.5 GHz	17.25 dBm
33 GHz	14.1 dBm
40 GHz	13.75 dBm

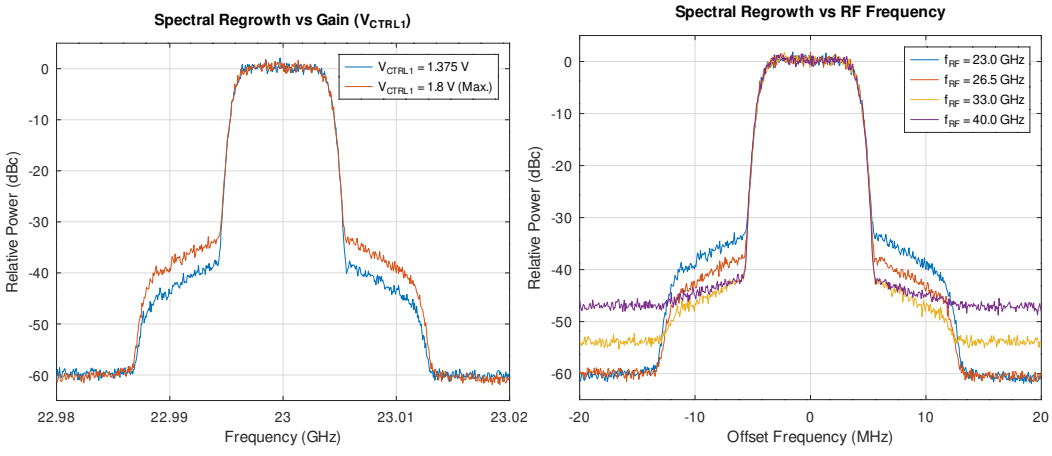
With the aforementioned modulated IF carrier input, the ACPR of the output signal was measured, with results presented in table 6.6. At 23 GHz, two measurements

were performed, each with a different gain, controlled by the voltage at the V_{CTRL1} pin of the ADMV1013. At this frequency, it is possible to see a degradation of 1.5 dB in the ACPR when the gain is at its maximum ($V_{CTRL1} = 1.8$ V), compared to a 3 dB gain back-off ($V_{CTRL1} = 1.375$ V). It is important to note that the ACPR of the input IF signal was -36.5 dBc, determined using setup 1 of fig. 6.9, meaning that it is not possible to obtain better results for either the up or downconverters. The spectrum of the signals used to measure the ACPR performance of the upconverter is depicted in fig. 6.12, where fig. 6.12a shows the variation of the spectral regrowth with gain (V_{CTRL1}) and fig. 6.12b traduces the performance at different RF carrier frequencies.

Table 6.6: Summary of the ACPR of a 10 MHz 16-QAM modulated carrier output from the mmWave expansion card transmitter.

Frequency	**Gain (V_{CTRL1})[1]**	**Output Power**	**ACPR**
23 GHz	1.375 V	0.8 dBm	-34.5 dBc
	1.8 V	3.13 dBm	-33 dBc
26.5 GHz	1.8 V	2.9 dBm	-35 dBc
33 GHz	1.8 V	-2.8 dBm	-35.5 dBc
40 GHz	1.8 V	-3.8 dBm	-35.5 dBc

[1] $V_{CTRL2} = 1.8$ V - maximum value



(a) (b)

Figure 6.12: mmWave upconverter output spectrum, showcasing spectral regrowth of a QAM modulated carrier: (b) vs RF frequency; (b) vs gain (V_{CTRL1}).

By leveraging setup 3, depicted in fig. 6.9, the evaluation of the mmWave receiver chain is performed. This setup is similar to setup 2, however, it comprises a loopback connection between the TX and RX RF connections. As the transmitter was

previously characterized, it is possible to de-embed it from the measurement, leaving only the receiver's performance, at least for the measured parameters presented here.

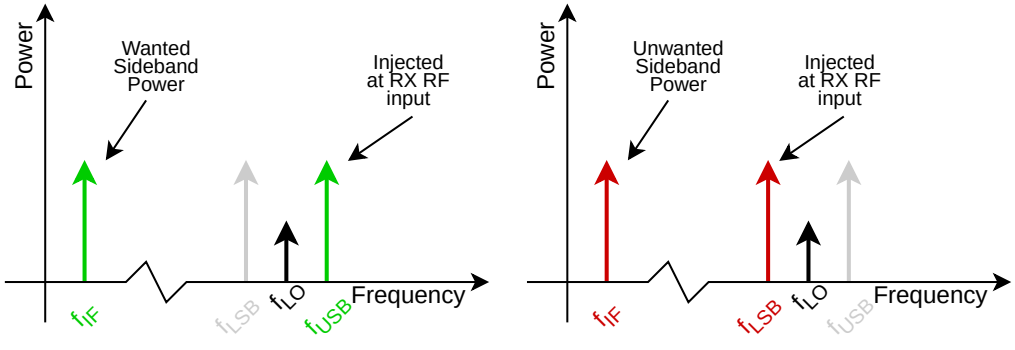
Similarly to the transmitter, the receiver conversion gain has been measured on the bench, at several RF and IF frequency combinations, with all the results compiled in table 6.7. It is, once again, possible to observe that the conversion gain is higher at an IF frequency of 2.5 GHz, and, as noted before, the gain variation across the measured IF frequency points is better than 2.5 dB. At 26.5 GHz, the receiver achieves the highest measured gain value of 14.7 dB.

Table 6.7: Summary of the mmWave expansion card receiver conversion gain at different RF and IF frequencies and maximum gain ($V_{CTRL} = 0$ V, control pin of the ADMV1014).

RF Frequency	**IF Frequency**	**Gain[1]**
23 GHz	3 GHz	12.6 dB
26.5 GHz	3 GHz	14.7 dB
33 GHz	3 GHz	9 dB
	2.5 GHz	10 dB
40 GHz	3 GHz	9.1 dB
	3.5 GHz	7.6 dB

[1] Digital tuners adjusted for maximum gain.

As important as the upconverter's image rejection, it is also relevant to measure this parameter in the receiver's downconverter, as this parameter is crucial to ensure only the wanted (upper) sideband is downconverted to IF. The transmitter image rejection measurement process is quite simple, as the relevant tones are all visible in the spectrum at the same time, due to the mix-up operation performed by the transmit mixer. In the downconverter, however, it is not possible to discern the desired downconverted sideband ($f_{LO} + f_{IF}$) from its image ($f_{LO} - f_{IF}$) in the spectrum analyzer. To overcome this limitation, the methodology depicted in fig. 6.13 was employed to indirectly obtain the image rejection. First, a tone with frequency $f_{LO} + f_{IF}$ is injected in the receiver's RF input port, and the power is measured at f_{IF}, from the IF output port, as depicted in fig. 6.13a. Then, the same procedure is followed, injecting instead a tone at $f_{LO} - f_{IF}$ and recording the power of the signal at f_{IF}, as shown in fig. 6.13b. By subtracting the measured values (in dBm), the image rejection of the receiver can be determined. The entire procedure that was just described was performed after tuning the digital control elements of



(a) (b)

Figure 6.13: Methodology for measuring the image rejection of the mmWave downconverter: (a) setup for measuring the wanted sideband power; (b) setup for measuring the unwanted sideband power.

the downconverter to achieve the best image rejection figure, and the results are presented in table 6.8. The downconverter presents an excellent image rejection of more than 57 dB at 23 GHz, and it is better than 45 dB for the entire set of measured combinations, which ensure proper attenuation of the unwanted sideband components.

Assessment of the receiver's distortion performance was performed by following the same procedure detailed previously for the transmitter, with the modulated carrier signal, however, by employing setup 3 of fig. 6.9, the receiver IF output was monitored instead of the TX RF output. It is worth noting that the achievable ACPR may not be better than -35.5 dBc, as this is the best figure obtained from the output of the transmitter. In table 6.9, the ACPR results of the receiver, obtained

Table 6.8: Summary of the mmWave expansion card receiver image rejection at different RF and IF frequencies and maximum gain (V_{CTRL} = 0 V).

RF Frequency	**IF Frequency**	**Rejection**[1]
23 GHz	3 GHz	57.1 dB
26.5 GHz	3 GHz	46.9 dB
33 GHz	3 GHz	51.4 dB
	2.5 GHz	55.25 dB
40 GHz	3 GHz	50.22 dB
	3.5 GHz	47.2 dB

[1] Digital tuners adjusted for maximum image rejection.

Table 6.9: Summary of the ACPR of a 10 MHz 16-QAM modulated carrier, at $f_{RF} = 26.5$ GHz and $f_{IF} = 3$ GHz and maximum gain ($V_{CTRL} = 0$ V), output from the mmWave expansion card receiver.

IF Output Power	**ACPR**
-17.9 dBm	-35 dBc
-10.6 dBm	-35 dBc
-1.7 dBm	-35 dBc
3.2 dBm	-26 dBc

at an RF and IF frequencies of 26.5 GHz and 3 GHz, respectively. It is possible to observe in fig. 6.14 that little to no distortion is introduced by the receiver when operated with an input power lower than -25.5 dBm. When operated at higher power levels, the internal amplifiers of the downconverter start to enter saturation, causing spectral regrowth of the signal. At an IF output power, however, the receiver still achieves an acceptable ACPR of -26 dBc. To conclude, the bench setup was leveraged to obtain the results in fig. 6.15, which represent the spectral contents of the downconverter's IF output up to 50 GHz. Besides the desired tone at 3 GHz (f_{IF}), a direct result of the downconversion from 40 GHz (f_{RF}), two other tones stand out of the noise floor of the instrument. A tone at 9.25 GHz, represents leakage of the divided LO signal, which is input to the IC, and, at 18.5 GHz, the second harmonic of the 9.25 GHz tone. Despite presenting power levels above the spectrum analyzer's noise floor, their power is more than 35 dB below the desired

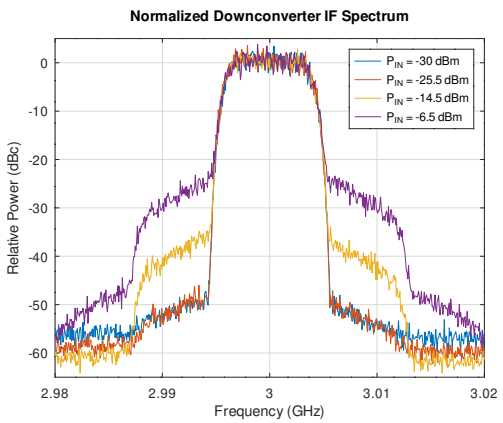


Figure 6.14: mmWave downconverter output spectrum, showcasing spectral regrowth of a QAM modulated carrier vs RF input power.

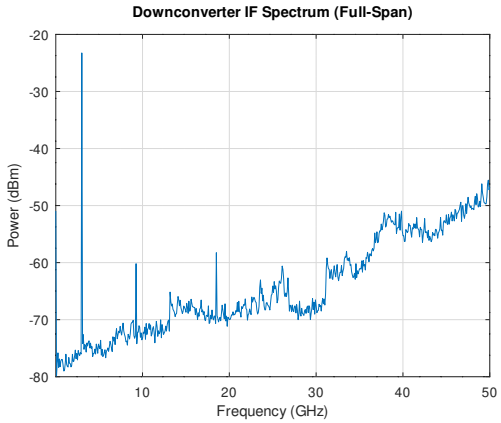


Figure 6.15: Full span spectrum of the mmWave downconverter IF output.

IF output tone. This rejection is not only provided by the downconverter IC itself, but also by the BPF employed on the IF output.

To assess the impedance matching of both RF and IF ports of the mmWave expansion card up and downconverters, an Agilent E8361A VNA was connected to the card, as depicted in the connection setups 1 and 2 of fig. 6.16. Figure 6.17

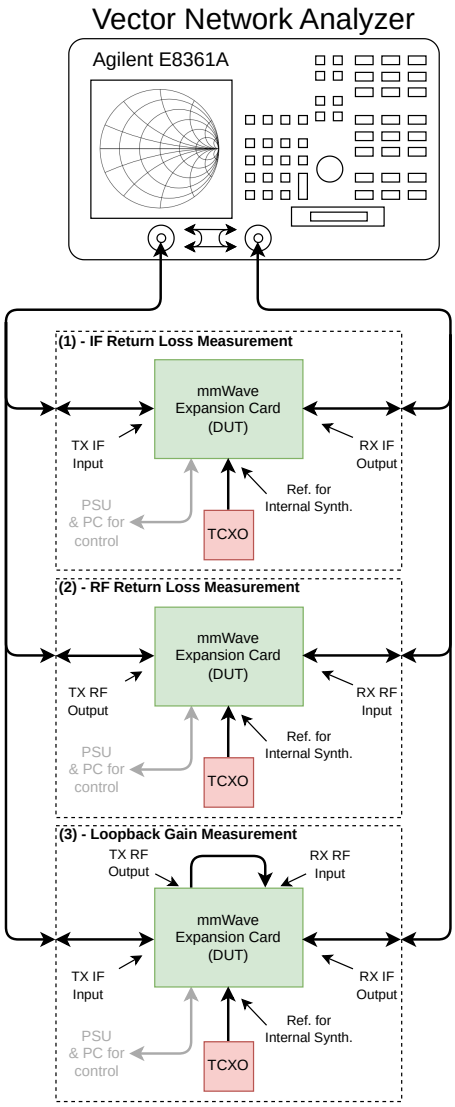


Figure 6.16: Instruments and connection diagram of the bench setup for testing the mmWave expansion card RF and IF ports return loss and loopback conversion gain.

traduces the IF ports' return loss obtained with setup 1 of fig. 6.16, which generally follow the return loss performance of the IF BPF filter, previously presented in fig. 5.4, as expected. It stays below -10 dB for almost the entire 2.5 GHz to 3.5 GHz operating band, slightly peaking above -10 dB near 3 GHz. In fig. 6.18, the return loss of the RF ports is presented, which, essentially, directly represents the impedance match of the converters' RF port. The transmitter does present a good

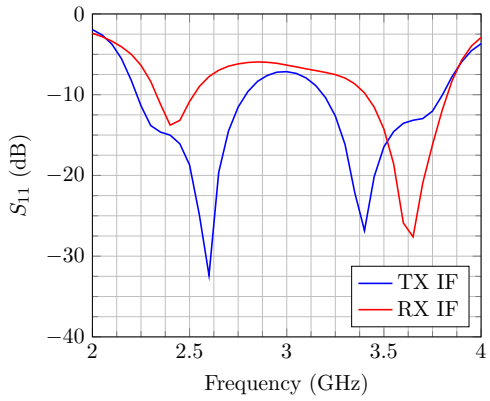


Figure 6.17: mmWave card measured IF ports return loss (S_{11}).

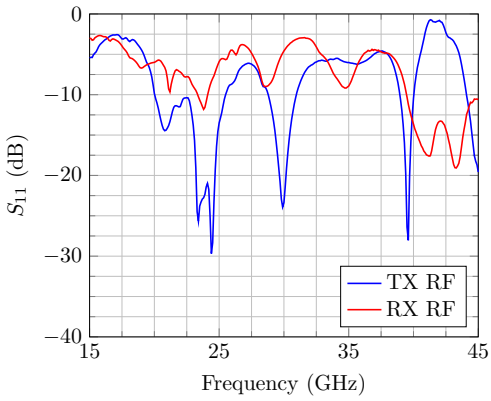


Figure 6.18: mmWave card measured RF ports return loss (S_{11}).

impedance match of better than -10 dB in some frequency bands such as in the 20 GHz to 25 GHz, 30 GHz and 40 GHz bands. In some bands, however, the impedance match can rise above -5 dB. As explained in section 5.2.1, the RF connections of the converters was directly routed to high-frequency (2.92 mm) connectors, in order to not introduce additional losses to the mmWave signals. It is not trivial to achieve perfect impedance matching across such a wide range of frequencies, especially when trying to design LNAs for minimum noise figure, as is the case in the mmWave downconverter/receiver. Thus, it is suggested that the SDR user further optimizes the antennas (or other connected equipment) for the impedance of the RF ports.

Using setup 3 of fig. 6.16, whose bench is depicted in fig. 6.19, the VNA was used to measure the loopback gain of the cascaded transmitter and receiver chains, at maximum, medium, and minimum gain levels. The maximum gain level corresponds

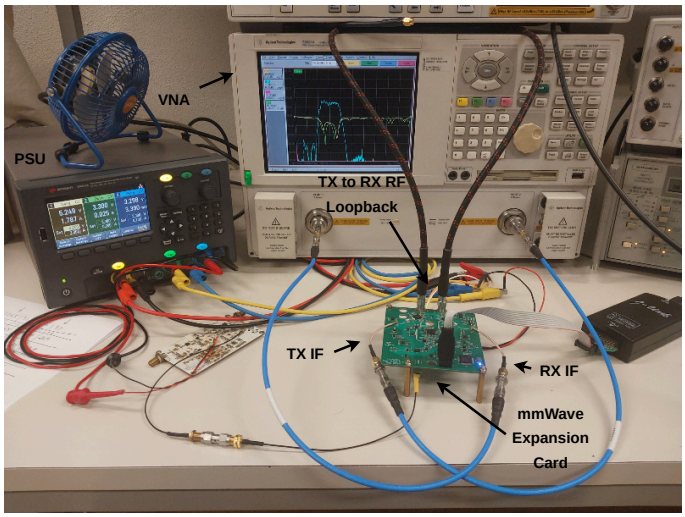


Figure 6.19: Bench setup for testing the mmWave expansion card RF and IF ports return loss and loopback conversion gain, using a VNA.

to all gain control voltages set to the maximum gain values, that is, TX V_{CTRL1} and V_{CTRL2} set to 1.8 V and RX V_{CTRL} set to 0 V. The medium gain level is represented by all aforementioned control voltages being set to 0.9 V, the center point of the gain scale. Finally, the minimum gain curves represent the gain of the cascaded chain with TX V_{CTRL1} and V_{CTRL2} set to 0 V and RX V_{CTRL} set to 1.8 V. The results of this measurement at different frequencies are presented in figs. 6.20 to 6.23. In all figures, the effect of the IF BPF can clearly be observed, as well as the effect of the control voltages on the overall gain.

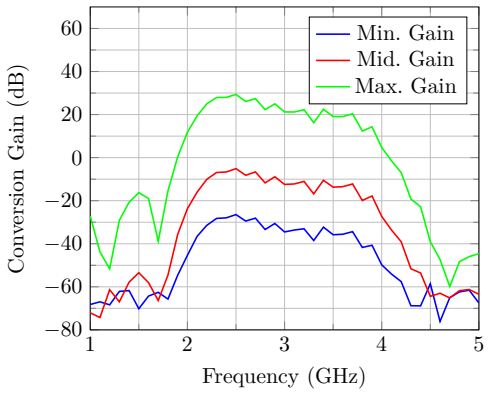


Figure 6.20: mmWave expansion card full chain (TX + RX) conversion gain at f_{RF} = 20 GHz and different gain settings.

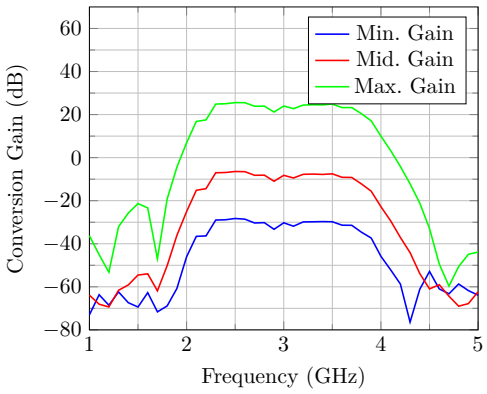


Figure 6.21: mmWave expansion card full chain (TX + RX) conversion gain at f_{RF} = 23.5 GHz and different gain settings.

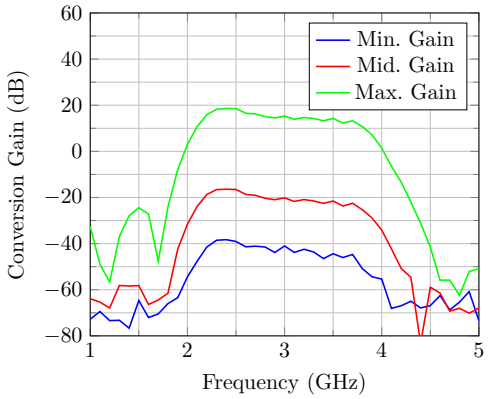


Figure 6.22: mmWave expansion card full chain (TX + RX) conversion gain at f_{RF} = 30 GHz and different gain settings.

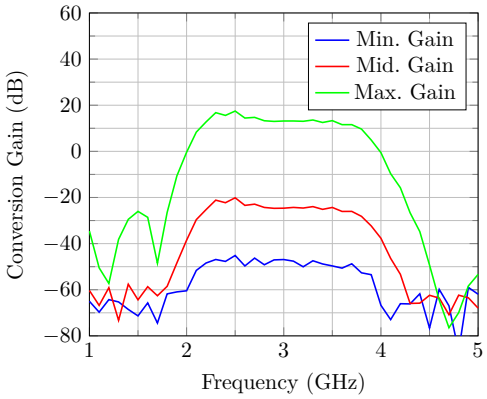


Figure 6.23: mmWave expansion card full chain (TX + RX) conversion gain at f_{RF} = 37 GHz and different gain settings.

By using the Agilent E4448A spectrum analyzer's phase noise analysis feature, and with the bench setup depicted in fig. 6.24, the phase noise performance of the mmWave expansion card's internal LO synthesizer was assessed. As mentioned in sections 3.2.3 and 5.2.2, with the externally implemented loop filter, it is possible to

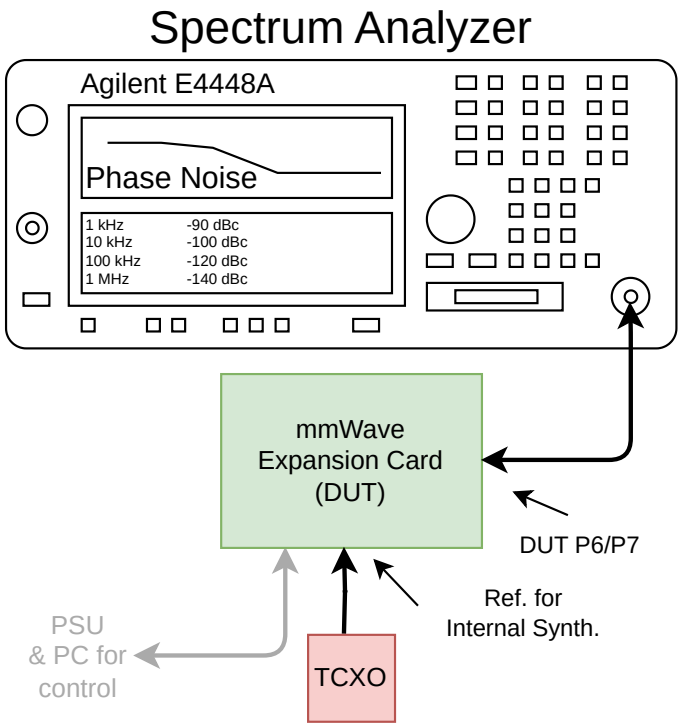


Figure 6.24: Instruments and connection diagram of the bench setup for testing the mmWave expansion card internal LO synthesizer phase noise.

alter the PLL's loop bandwidth using the digitally controllable charge pump current. This feature was leveraged to obtain phase noise results for three different loop bandwidth (f_c) configurations: 10 kHz, 50 kHz and 100 kHz. Figures 6.25a, 6.25c and 6.25e traduce the phase noise of the synthesizer at 5 GHz, figs. 6.25b, 6.25d and 6.25f at 7.5 GHz, figs. 6.26a, 6.26c and 6.26e at 9.25 GHz and figs. 6.26b, 6.26d and 6.26f at 10.25 GHz. In all figures, it is possible to observe the clear difference on the phase noise profile caused by the three different loop bandwidth configurations. When the synthesizer is configured with a 10 kHz loop bandwidth (figs. 6.25a, 6.25b, 6.26a and 6.26b), it is possible to observe a slight peaking in the phase noise profile of the synthesizer's output. This behavior is due to the loop bandwidth ($f_c = 10$ kHz) being close to the zero frequency, f_z, which, with the currently implemented loop filter, is approximately 8 kHz. As referred in section 3.2.3, f_c should not be inferior to f_z, and, as the PLL configuration approaches the point at which this condition is violated, phase noise peaking starts to occur, leading to potential instability in the system [48]. Is is also possible to observe that the loop bandwidth configuration involves a trade-off between close-in and far-out phase noise performance, as within the loop bandwidth ($f < f_c$), the majority of the phase noise is originated in the PLL reference, whereas outside of the loop bandwidth ($f > f_c$), the VCO dominates the phase noise performance of the entire PLL [76].

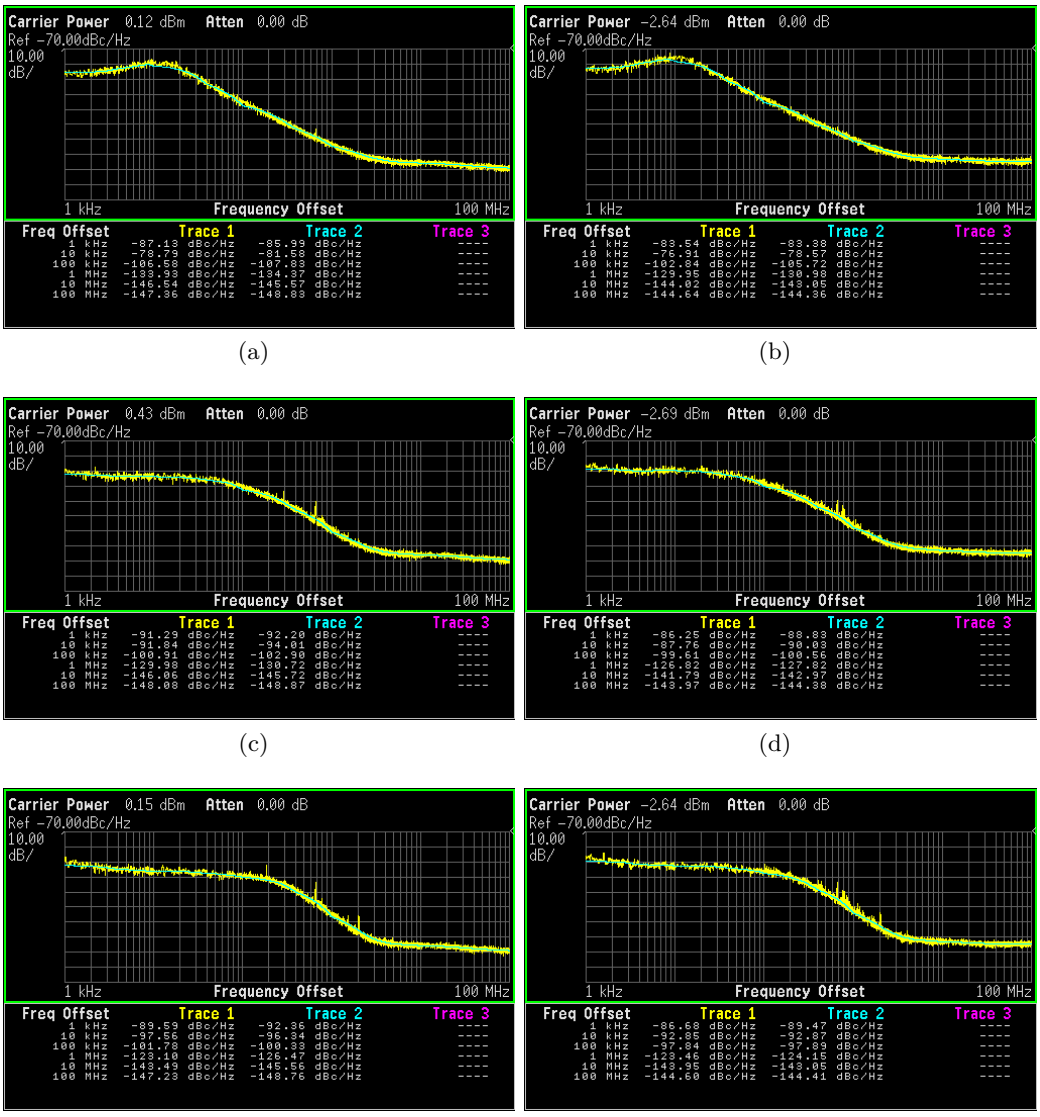


Figure 6.25: Measured phase noise of the internal synthesizer of the mmWave expansion card: (a) at f_{OUT} = 5 GHz and f_c = 10 kHz; (b) at f_{OUT} = 7.5 GHz and f_c = 10 kHz; (c) at f_{OUT} = 5 GHz and f_c = 50 kHz; (d) at f_{OUT} = 7.5 GHz and f_c = 50 kHz; (e) at f_{OUT} = 5 GHz and f_c = 100 kHz; (f) at f_{OUT} = 7.5 GHz and f_c = 100 kHz.

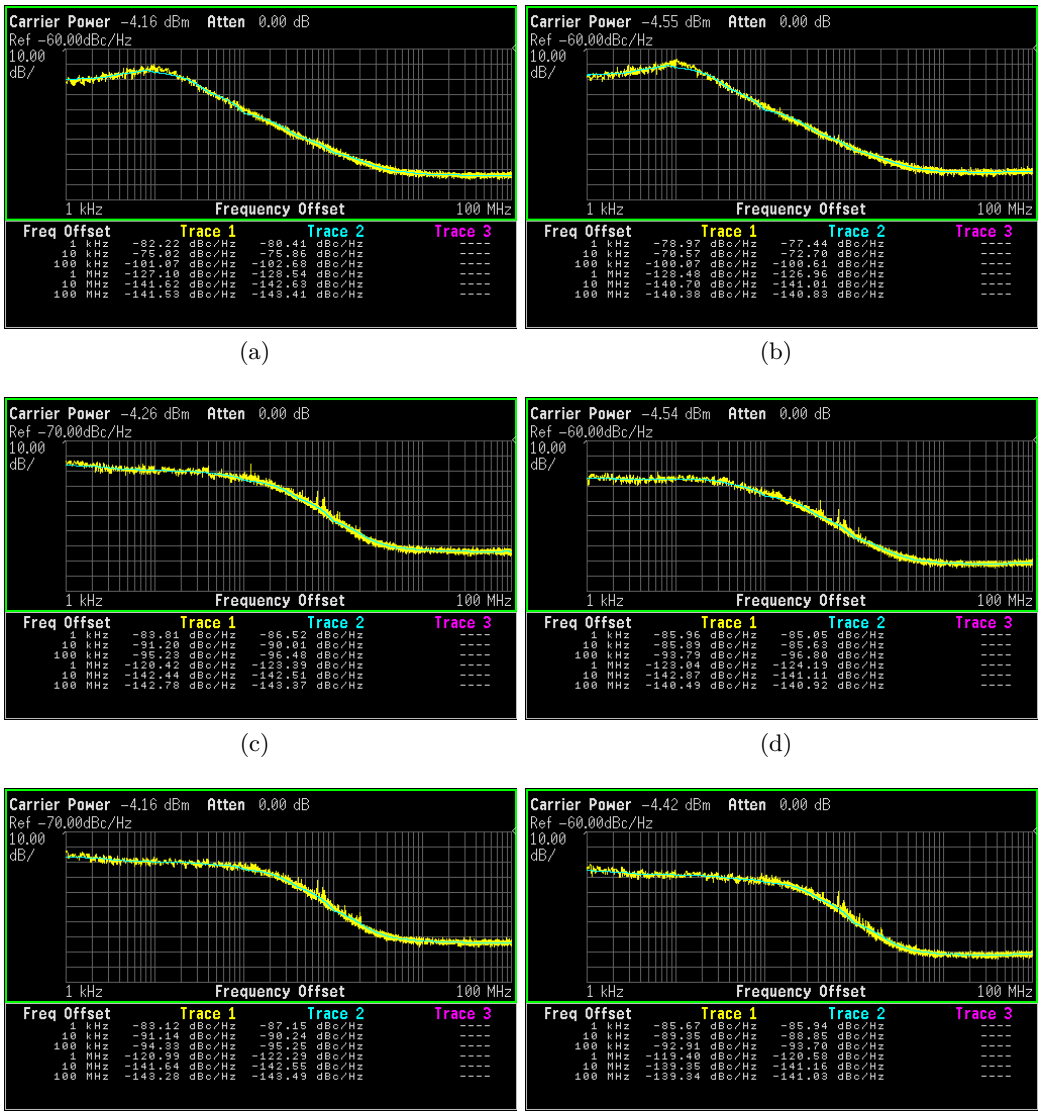


Figure 6.26: Measured phase noise of the internal synthesizer of the mmWave expansion card: (a) at $f_{OUT} = 9.25$ GHz and $f_c = 10$ kHz; (b) at $f_{OUT} = 10.25$ GHz and $f_c = 10$ kHz; (c) at $f_{OUT} = 9.25$ GHz and $f_c = 50$ kHz; (d) at $f_{OUT} = 10.25$ GHz and $f_c = 50$ kHz; (e) at $f_{OUT} = 9.25$ GHz and $f_c = 100$ kHz; (f) at $f_{OUT} = 10.25$ GHz and $f_c = 100$ kHz.

As mentioned in section 5.4, the RF power sensors utilized to measure the LO signal power were calibrated across frequency. Figure 6.27 depicts the voltage read by the MCU ADC across LO frequency, at three different synthesizer power levels. As the synthesizer provides twelve discrete steps of power control, a measurement was performed at levels 1 (minimum), 6 (mid-scale) and 12 (maximum). Through this curve, and correlating with the actual LO power level measured with the spectrum analyzer, the control system of the mmWave expansion card can accurately manage the LO power level, ensuring the correct LO power reaches the mmWave up and downconverter ICs.

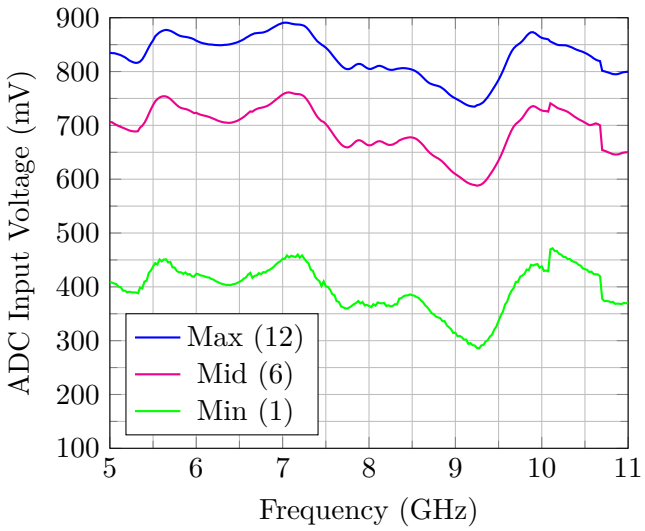


Figure 6.27: Voltage at the input of the MCU of the mmWave expansion card vs LO frequency at different internal synthesizer power levels.

6.3 DEMONSTRATION APPLICATIONS

As presented in section 4.4.4, a set of transmitter and receiver demonstration applications were developed with the aim of exemplifying the use of the SoapySDR library to interface with the SDR. These applications were also levraged in the laboratory to showcase the usage of the complete SDR platform, achieving over-the-air transmission of a live video feed from a digital camera. Two scenarios were tested, covering the base SDR platform, with a sub 6 GHz modulated carrier, and the complete mmWave SDR, utilizing a similarly modulated carrier centered at 40 GHz.

A transmitter configuration, depicted in fig. 6.28, was assembled on a supporting platform on wheels, for ease of transportation. This configuration comprises the

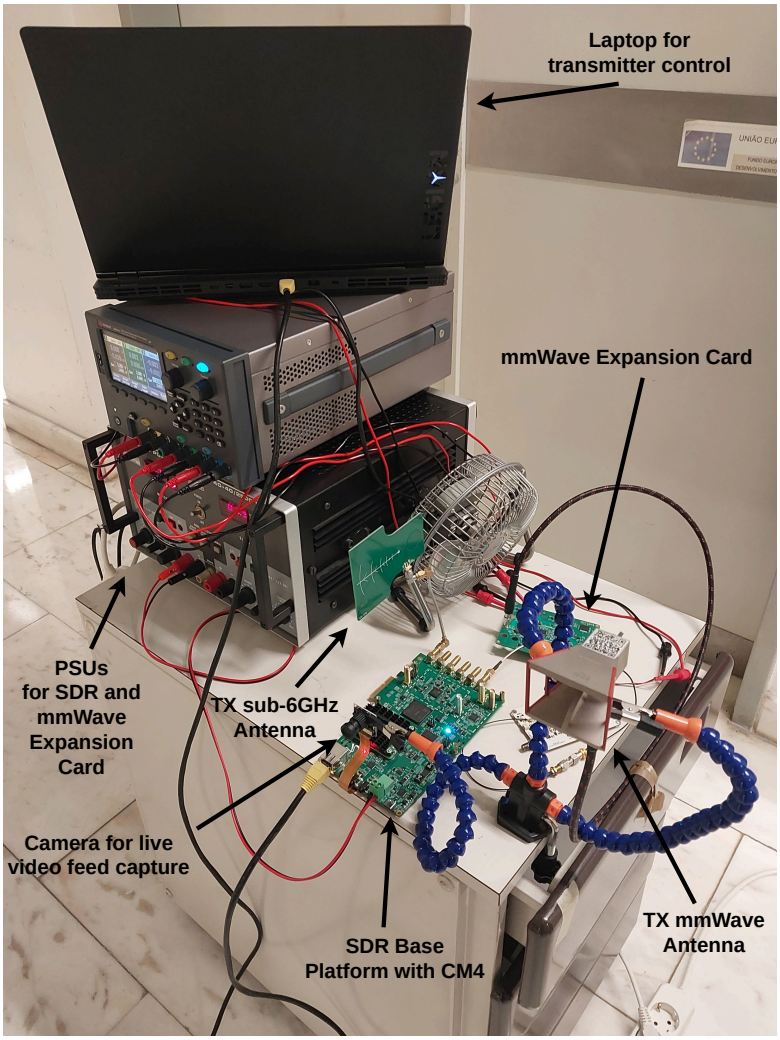


Figure 6.28: Transmitter cart setup for a live video streaming demonstration using the SDR.

base SDR platform with a CM4 installed and a Vivaldi antenna working from 2.4 GHz to 10 GHz, which is used to transmit the sub 6 GHz modulated carrier. A mmWave expansion card is also included in the transmitter setup, whose IF input is connected to the base SDR platform. The RF output of the mmWave expansion card is connected to an RF SPIN DRH40 double ridged horn antenna, operating at up to 40 GHz, used for the mmWave transmission demonstration. Power supplies for the SDR are also included in the transmitter cart, as can be seen in fig. 6.28, as well as a laptop, connected via Ethernet to the base SDR, for controlling the transmitter.

A second movable cart was used to assemble a receiving setup, with a host PC, accompanying monitor and an SDR installed on a PCIe slot. Figure 6.29 depicts this receiver setup, prepared with a Vivaldi antenna similar to the one used on the transmitter, for reception of the sub 6 GHz modulated carrier. In this figure,

the carrier can be observed in the spectrum monitor and corresponding waterfall, generated using the GQRX software [77]. This software uses GNU Radio [14] as its core and offers an intuitive interface which allows monitoring the spectrum of a connected SDR, among other basic features. This software is able to interface with SoapySDR compatible SDRs, proving that the developed SoapySDR driver works as intended. Similarly, fig. 6.30 depicts the same setup, but with a mmWave

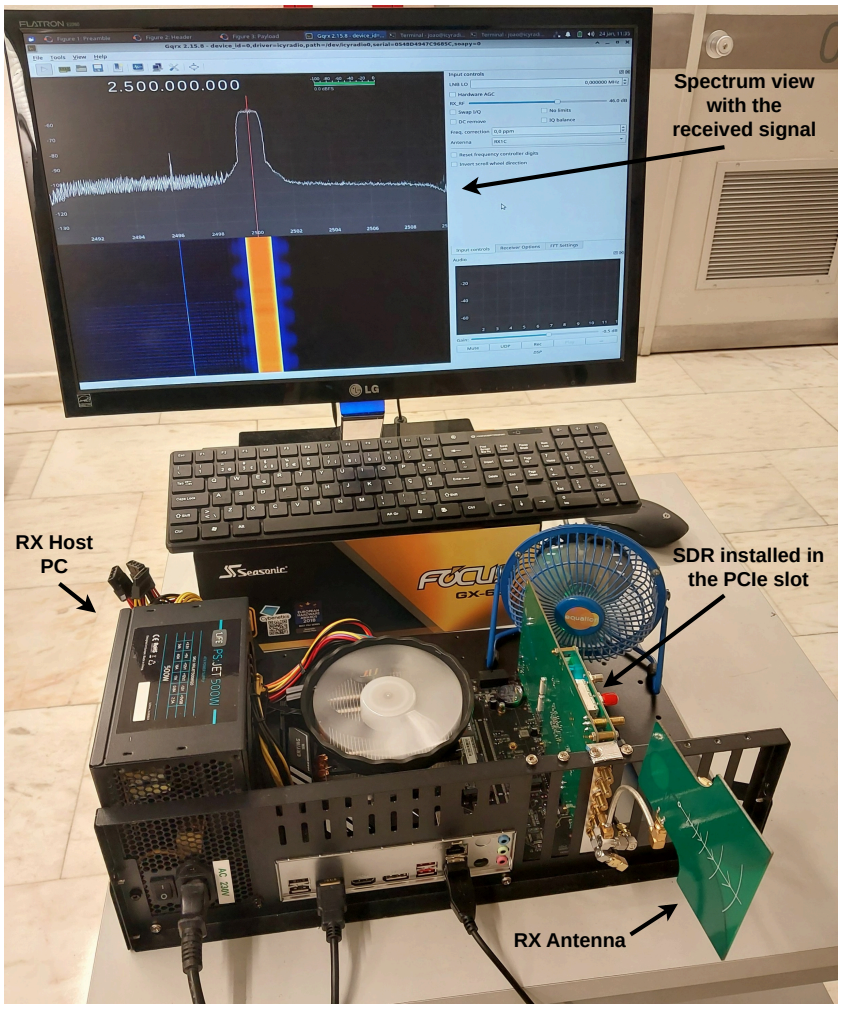


Figure 6.29: Receiver cart setup for a live video streaming demonstration using the SDR at sub 6 GHz bands.

horn antenna, again, similar to the one employed at the mmWave transmitter, to receive the 40 GHz modulated carrier via the mmWave expansion card, which is inserted onto the base SDR platform. In fig. 6.30 it is possible to observe the video stream being played back on the PC's monitor, as well as the received 64-QAM signal constellation, also depicted in fig. 6.31b.

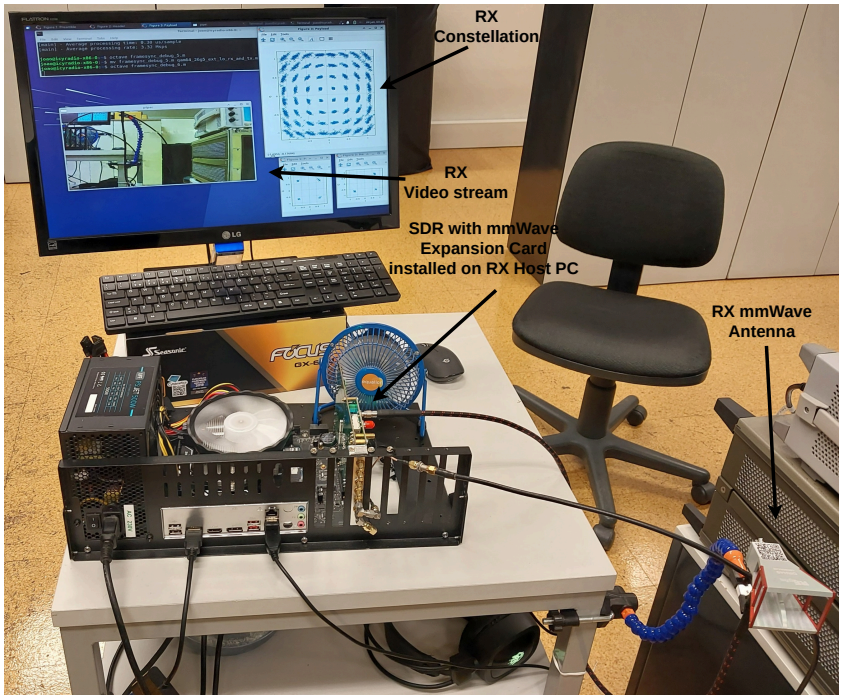


Figure 6.30: Receiver cart setup for a live video streaming demonstration using the SDR at mmWave frequencies.

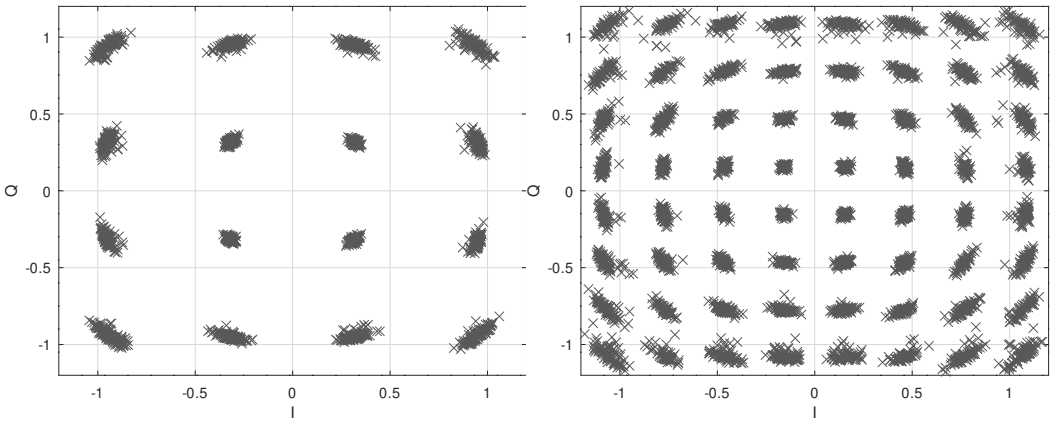


Figure 6.31: Base SDR platform demodulated signal constellation: (a) 16-QAM; (b) 64-QAM.

7

CONCLUSIONS AND FUTURE WORK

Throughout this work, the design, implementation, and evaluation of a cutting-edge SDR platform that pushes the boundaries of flexibility, performance, and expandability in wireless communication systems have been explored. The literature review was first presented in chapter 2. This foundation was crucial in understanding the state-of-the-art in digitally intensive RF transceivers and allowed informed design choices to be made during the SDR architecture definition, also presented in chapter 2, and throughout the following development process.

7.1 CONCLUSIONS

The circuit development of the SDR base platform, a critical phase that translated the SDR topology into practical hardware, was presented in chapter 3. Each functional block was meticulously designed, including the signal processing core, RF transceiver, mmWave synthesizer, clock management, and power management systems. Special attention was given to the expansion card interconnections, a feature that significantly enhances the platform's modularity and future-proofing. The chapter concluded with a discussion on the PCB stackup and layout, highlighting the challenges and solutions in physically implementing the designed circuits. The hardware development was complemented by extensive firmware and software development, as outlined in chapter 4. This chapter explored the FPGA design architecture, detailing both custom and proprietary IP cores that form the heart of the SDR's digital processing capabilities. Development of custom IP cores was addressed, as well as their interconnection with several other proprietary IP cores. The software section covered various aspects from low-level firmware to high-level user applications, ensuring a complete and user-friendly SDR ecosystem. The development of the mmWave expansion card was detailed in chapter 5. This card extends the capabilities of the SDR platform into the millimeter-wave frequency range, opening up new possibilities for research and applications in 5G and beyond. The chapter provided an in-depth look at the system design considerations, circuit design, and performance estimations of this expansion card. The integration of this

card with the base platform demonstrates the success of the proposed modular design approach and the platform's expandability.

The laboratory and real environment assessment of the SDR base platform, the mmWave expansion card and the mmWave SDR, which comprises both the SDR base platform and the mmWave card, interconnected together, were performed and presented in chapter 6, providing empirical evidence of the system's capabilities. Through a series of tests and demonstrations, it was proved that the SDR platform meets and, in some cases, exceeds the initial design specifications. The frequency range of the base platform, spanning from 100 MHz to 6 GHz, provides excellent coverage for a wide range of applications. Moreover, the successful implementation of the mmWave expansion card, operating from 24 GHz to 40 GHz, demonstrates the platform's expandability and readiness for next-generation wireless technologies. The achieved bandwidth of 56 MHz meets the initial specification, providing ample room for wideband signal processing and high data rate communications. The implementation of MIMO 2x2 capability, with each TX/RX channel having two digitally selectable input/output signal paths, offers flexibility in handling different frequency bands and enhances the platform's versatility. The ability to operate from 500 ksps to over 60 MSPS, with the potential to reach over 120 MSPS when exploiting undocumented features of the RF Transceiver, exceeds the originally set specifications. This wide range of sampling rates makes this platform suitable for a diverse array of applications, from narrowband IoT to wideband 5G communications.

A unique feature of the SDR is the inclusion of an on-board frequency synthesizer operating from 170 MHz to 18 GHz. This wide-range synthesizer not only supports the base platform's operations but also provides a valuable resource for generating local oscillator signals for daughter cards, including the mmWave expansion card. This feature significantly enhances the platform's expandability and future-proofing.

Frequency accuracy is a critical parameter of any SDR system, and the developed platform achieves a respectable 0.5 ppm accuracy. Furthermore, the system was designed with the flexibility to easily connect an external reference, allowing for even greater accuracy when required. This feature ensures that the SDR can meet the stringent timing requirements of various wireless standards and experimental setups.

The processing capabilities of the SDR platform are another relevant performance metric. By incorporating both FPGA-based hardware processing and a powerful quad-core ARM Cortex-A72 CPU running at 2.1 GHz, it was possible to conceive a system capable of handling complex signal processing tasks with ease. This

mixed approach to processing allows for optimal task distribution, with time-critical operations handled in hardware while more complex, higher-level tasks can be managed in software. Data offloading, often a bottleneck in SDR systems, has been addressed through the usage of a PCIe interface capable of speeds up to 8 Gbps. This high-speed interface ensures that the developed and implemented platform can handle the data rates required for advanced wireless protocols and research applications without becoming a limiting factor.

Flexibility in power options was another design goal that has been successfully met. The SDR can be powered via a standard DC input ranging from 9 V to 24 V, USB PD, or directly from a PCIe slot. This versatility in power options enhances the platform's usability across different scenarios, from laboratory setups to field deployments. The over-provision of the power system ensures that the platform is ready to support expansion cards that may have higher power requirements, further extending the life and utility of the base system.

The developed SDR platform represents a significant advancement in the field of software defined radios. Its combination of wide frequency range, high bandwidth, flexible processing options, and expandability makes it a powerful tool for researchers and developers in wireless communications. The integration of the mmWave expansion card demonstrates the platform's readiness for next-generation wireless technologies, including 5G and beyond. The adopted modular design approach ensures that this SDR platform will remain relevant and useful for years to come. As new wireless technologies emerge, the platform can be adapted and expanded to overcome new challenges. The over-provisioned power system and flexible expansion options provide a solid foundation for future developments.

Aside from achieving the primary objectives, this project has also opened up several future research and development opportunities. In the broader context of wireless communications research, the SDR platform provides a valuable tool for exploring new modulation schemes, investigating spectrum sharing techniques, and developing novel wireless protocols. Its flexibility and performance make it suitable for a wide range of applications, from IoT, advanced RFID systems, and fixed/mobile cellular communications to radar systems and beyond.

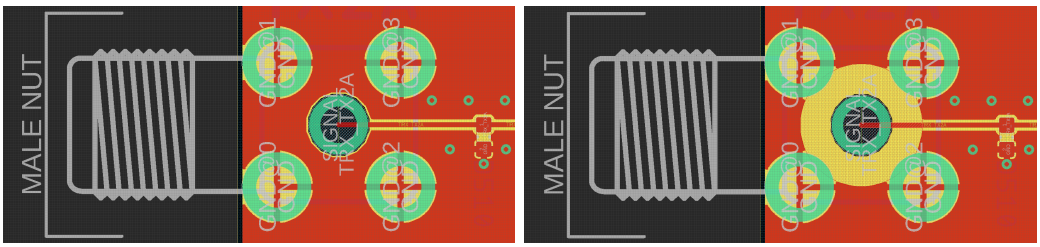
7.2 FUTURE WORK

As previously mentioned, the successful development and implementation of this SDR platform have opened up various future R&D use cases. This section outlines

potential areas for improvement and expansion, ensuring that the platform continues to evolve and remain at the forefront of wireless communication technology.

7.2.1 *Refinement of the SDR base platform*

While the current base SDR platform has demonstrated excellent performance, there are minor hardware issues that require attention. Specifically, the RF output connectors' footprint and the synchronization unit's phase noise performance need to be addressed. These refinements, though small, are crucial for enhancing the overall reliability and usability of the platform. An extensive analysis of the RF connectors footprint on the base SDR PCB, through 3D EM simulations, has shown that the current implementation may impact RF performance, namely the impedance matching and power output. Figure 7.1 showcases the issue and proposed solution, which involves retracting the ground plane around the center contact of the connectors.



(a) (b)

Figure 7.1: Rectification of the RF connectors: (a) before rectification; (b) after rectification.

This rectified footprint has shown better performance in simulation, and will hopefully improve signal integrity and reduce potential losses, particularly at higher frequencies. This enhancement is critical for maintaining the platform's performance across its entire operational frequency range.

The current synchronization unit architecture ensures a highly flexible clock distribution design, which aligns with the initial goals set for the platform. However, the current implementation does not offer optimal phase noise performance. Thus, this section of the base SDR platform needs to be refined in order to achieve an even better performance. By rectifying these issues, it can be ensured that the SDR provides a more robust and dependable foundation for various applications and research projects.

7.2.2 *Base SDR platform improvements*

Although the current development approach for the mmWave SDR follows traditional techniques, it is of paramount importance to gain insight and experience into the field of mmWave SDR design. This will enable the evolution of SDRs based on sampling techniques, with the aim of first replacing the RF section with RF DACs and ADCs, as explained in sections 2.2 and 2.3, followed by the replacement of the mmWave module with ultra-fast RF DACs and ADCs, when economically viable.

This idea aims to evaluate the performance benefits of direct RF sampling and generation. The use of high-speed converters could enhance the platform's flexibility, allowing it to operate over a wider frequency range without the need for multiple analog front-end designs. This approach aligns well with the software-defined nature of the platform, potentially enabling more of the signal chain to be implemented in the digital domain. Nevertheless, the compatibility with existing expansion cards should be guaranteed, ensuring the modularity of the platform.

7.2.3 *Development of additional expansion cards*

To further extend the capabilities and applications of the developed SDR platform, several proposals for the development of new expansion cards arise:

- *X-Band Expansion Card:* This card would enable operation in the 8-12 GHz frequency range (possibly extending the lower end to 6 GHz, to provide a contiguous operating range), opening up possibilities for satellite communication, radar systems, and other applications in this band;
- *Ku-Band expansion card:* Operating in the 12-18 GHz range, this expansion would be valuable for satellite communications, particularly for broadcast or amateur satellite services. Extending the card's operating range up to 20 GHz could help close the gap with the developed mmWave expansion card, ensuring the SDR platform's contiguous operation from 100 MHz to 40 GHz;
- *High-Power 5G FR1 expansion card:* Focusing on the 3.6 GHz band, this card would incorporate a high-power amplifier suitable for 5G FR1 base-station applications. This expansion would allow the SDR system to be used in more realistic cellular network research and prototyping scenarios.

7.2.4 *Enhanced software and firmware development*

While not explicitly mentioned in the provided future work topics, it's important to consider ongoing software and firmware development to fully leverage the hardware capabilities of the SDR platform and its expansions. This could include the development of optimized signal processing algorithms for specific applications (e.g., 5G NR, radar processing, satellite communications). The creation of user-friendly interfaces and software tools to simplify the configuration and operation of the SDR across various use cases is also a relevant topic worth exploring. Further software advanced features such as cognitive radio capabilities, dynamic spectrum access, multi-standard operation, and digital pre-distortion may also be explored.

BIBLIOGRAPHY

[1] José Machado Fernández. "Software Defined Radio: Basic Principles and Applications". In: *Revista Facultad de Ingenier'ia* 24 (Jan. 2015), pp. 79–96. DOI: 10.19053/01211129.3160.

[2] Michael L. Dickens, Brian P. Dunn, and Laneman J. Nicholas. "Design and Implementation of a Portable Software Radio". In: *IEEE Communications Magazine* 46.8 (2008), pp. 58–66. DOI: 10.1109/MCOM.2008.4597105.

[3] J. Mitola. "Software radios-survey, critical evaluation and future directions". In: *[Proceedings] NTC-92: National Telesystems Conference*. 1992, pp. 13/15–13/23. DOI: 10.1109/NTC.1992.267870.

[4] R.I. Lackey and D.W. Upmal. "Speakeasy: the military software radio". In: *IEEE Communications Magazine* 33.5 (1995), pp. 56–61. DOI: 10.1109/35.392998.

[5] Robert Bogdan Staszewski. "Digitally intensive wireless transceivers". In: *IEEE Design & Test of Computers* 29.6 (2012), pp. 7–18. DOI: 10.1109/MDT.2012.2209392.

[6] Yoshimi Fujii et al. "28GHz Cooperative Digital Beamforming for 5G Advanced System on an SDR Platform". In: *2021 IEEE Radio and Wireless Symposium (RWS)*. 2021, pp. 80–82. DOI: 10.1109/RWS50353.2021.9360368.

[7] Martin Danneberg et al. "USRP-Based Platform for 26/28 GHz mmWave Experimentation". In: *2020 IEEE Wireless Communications and Networking Conference Workshops (WCNCW)*. 2020, pp. 1–6. DOI: 10.1109/WCNCW48565.2020.9124792.

[8] Kang Wang et al. "SDR Implementation of an End-to-End mmWave Testbed Based on Phased Antenna Array". In: *2019 11th International Conference on Wireless Communications and Signal Processing (WCSP)*. 2019, pp. 1–6. DOI: 10.1109/WCSP.2019.8928067.

[9] Tayfun Nesimoglu. "A review of Software Defined Radio enabling technologies". In: *2010 10th Mediterranean Microwave Symposium*. 2010, pp. 87–90. DOI: 10.1109/MMW.2010.5605145.

[10] Thangadurai Natarajan and Chetna Kh. "A Review on Recent Trends in Software Defined Radio Design and Applications". In: *International Journal of Advanced Research in Electronics and Communication Engineering (IJARECE)* 6 (Oct. 2017), pp. 1021–1025.

[11] Robert Keim. *Introduction to software-defined radio - technical articles*. Feb. 2017. URL: https://www.allaboutcircuits.com/technical-articles/introduction-to-software-defined-radio.

[12] Rami Akeela and Behnam Dezfouli. "Software-defined Radios: Architecture, State-of-the-art, and Challenges". In: *CoRR* abs/1804.06564 (2018). arXiv: 1804.06564. URL: http://arxiv.org/abs/1804.06564.

[13] András Retzler. *csdr*. https://github.com/ha7ilm/csdr.

[14] *GNU Radio*. https://www.gnuradio.org/.

[15] Joseph D. Gaeddert. *liquid-dsp*. https://github.com/jgaeddert/liquid-dsp.

[16] Pothosware. *SoapySDR*. https://github.com/pothosware/SoapySDR.

[17] *Software defined radio market size, share & trends*. Nov. 2022. URL: https://www.marketsandmarkets.com/Market-Reports/software-defined-radio-market-138946173.html.

[18] *Software Defined Radio Market Size, Share & Industry Analysis*. July 2021. URL: https://www.fortunebusinessinsights.com/software-defined-radios-market-102524.

[19] The Business Research Company. *Software defined radio growth forecast 2025-2034: Trends, opportunities, and key insights you need to know - latest global market insights*. Jan. 2025. URL: https://blog.tbrc.info/2025/01/software-defined-radio-market/.

[20] Ram Sunil Kanumalli et al. "Digitally-intensive transceivers for future mobile communications—emerging trends and challenges". In: *e & i Elektrotechnik und Informationstechnik* 135 (Feb. 2018). DOI: 10.1007/s00502-017-0576-1.

[21] E. Hogenauer. "An economical class of digital filters for decimation and interpolation". In: *IEEE Transactions on Acoustics, Speech, and Signal Processing* 29.2 (1981), pp. 155–162. DOI: 10.1109/TASSP.1981.1163535.

[22] David M Pozar. *Microwave Engineering*. en. 4th ed. Nashville, TN: John Wiley & Sons, Apr. 2012. ISBN: 9781118213636.

[23] Ariel Luzzatto and Gadi Shirazi. "Modern Transceiver Architectures". In: Jan. 2007, pp. 1–33. ISBN: 9780470060766. DOI: 10.1002/9780470060810.ch1.

[24] Umesh Jayamohan. *Not your grandfather's ADC: RF sampling adcs offer advantages in systems design*. July 2015. URL: https://www.analog.com/en/technical-articles/rf-sampling-adc-offer-advantages-in-systems-design.html.

[25] Daniel E. Fague. *New RF DAC Broadens Software-defined Radio Horizon*. July 2016. URL: https://www.analog.com/en/analog-dialogue/articles/new-rf-dac-broadens-sdr-horizon.html.

[26] Gil Engel et al. "A 16-bit 10Gsps current steering RF DAC in 65nm CMOS achieving 65dBc ACLR multi-carrier performance at 4.5GHz Fout". In: *2015 Symposium on VLSI Circuits (VLSI Circuits)*. 2015, pp. C166–C167. DOI: 10.1109/VLSIC.2015.7231252.

[27] Mickaël Dardaillon et al. "Software defined radio architecture survey for cognitive testbeds". In: *2012 8th International Wireless Communications and Mobile Computing Conference (IWCMC)*. 2012, pp. 189–194. DOI: 10.1109/IWCMC.2012.6314201.

[28] Peichuan Zhang et al. "Real-time signal processing for FM-based passive bistatic radar using GPUs". In: *2014 19th International Conference on Digital Signal Processing*. 2014, pp. 536–540. DOI: 10.1109/ICDSP.2014.6900723.

[29] RAPIDS - Open GPU Data Science. *cuSignal*. https://github.com/rapidsai/cusignal.

[30] CuPy - NumPy & SciPy for GPU. *CuPy*. https://github.com/cupy/cupy.

[31] Ian Kuon, Russell Tessier, and Jonathan Rose. 2008. DOI: 10.1561/1000000005.

[32] Philippe Coussy et al. "An Introduction to High-Level Synthesis". In: *IEEE Design & Test of Computers* 26.4 (2009), pp. 8–17. DOI: 10.1109/MDT.2009.69.

[33] L. R. Rabiner, B. Gold, and C. K. Yuen. "Theory and Application of Digital Signal Processing". In: *IEEE Transactions on Systems, Man, and Cybernetics* 8.2 (1978), pp. 146–146. DOI: 10.1109/TSMC.1978.4309918.

[34] Steven W. Smith. *The scientist and engineer's Guide to Digital Signal Processing*. California Technical Pub., 1997.

[35] Stephen A. Dyer and Brian K. Harms. "Digital Signal Processing". In: ed. by Marshall C. Yovits. Vol. 37. Advances in Computers. Elsevier, 1993, pp. 59–117. DOI: https://doi.org/10.1016/S0065-2458(08)60403-9. URL: https://www.sciencedirect.com/science/article/pii/S0065245808604039.

[36] A. Gatherer et al. “DSP-based architectures for mobile communications: past, present and future”. In: *IEEE Communications Magazine* 38.1 (2000), pp. 84–90. DOI: 10.1109/35.815456.

[37] Kasey Chatzopoulos, James Wong, and Murtaza Thahirally. *24 GHz to 44 GHz wideband integrated upconverters and downconverters boost microwave radio performance while reducing size*. Feb. 2023. URL: https://www.analog.com/en/resources/design-notes/wideband-integrated-upconverters-and-downconverters-boost-microwave-radio-performance.html.

[38] Alireza Zandieh et al. “128-GS/s ADC Front-End with Over 60-GHz Input Bandwidth in 22-nm Si/SiGe FDSOI CMOS”. In: *2018 IEEE BiCMOS and Compound Semiconductor Integrated Circuits and Technology Symposium (BCICTS)*. 2018, pp. 271–274. DOI: 10.1109/BCICTS.2018.8550842.

[39] *Per Vices Corporation*. https://www.pervices.com/.

[40] Analog Devices. *AD9361 datasheet*. https://www.analog.com/media/en/technical-documentation/data-sheets/ad9361.pdf.

[41] Analog Devices. *ADF4371 datasheet*. https://www.analog.com/media/en/technical-documentation/data-sheets/adf4371.pdf.

[42] Renesas Electronics. *8V97003 datasheet*. https://www.renesas.com/en/document/dst/8v97003-datasheet.

[43] Analog Devices. *ADMV1013 datasheet*. https://www.analog.com/media/en/technical-documentation/data-sheets/admv1013.pdf.

[44] Analog Devices. *ADMV1014 datasheet*. https://www.analog.com/media/en/technical-documentation/data-sheets/admv1014.pdf.

[45] R.E. Best. *Phase-locked Loops: Theory, Design, and Applications*. McGraw-Hill, 1984. ISBN: 9780070050501.

[46] Floyd Martin Gardner. *Phaselock techniques*. Wiley-Interscience, 2005. ISBN: 978-0-471-73268-6.

[47] Renesas Electronics. *8V97003 Performance Optimization Guidelines Application Note*. https://www.renesas.com/ja/document/apn/8v97003-performance-optimization-guidelines.

[48] Renesas Electronics. *PLL Loop Filter Design and Fine Tuning Application Note*. https://www.renesas.cn/zh/document/apn/pll-loop-filter-design-and-fine-tuning.

[49] Skyworks. *Si5351 datasheet*. https://www.skyworksinc.com/-/media/Skyworks/SL/documents/public/data-sheets/Si5351-B.pdf.

[50] Jim Williams. *Analog Circuit Design: Art, science, and personalities*. Butterworth-Heinemann, 2015. ISBN: 9781483102313.

[51] R. Ghiorse. *Monitoring and Sequencing in Multirail Power-Supply Systems*. Nov. 2011. URL: https://www.analog.com/en/resources/analog-dialogue/articles/monitoring-and-sequencing-multirail-power-supply-sys.html.

[52] Analog Devices. *ADM7171 datasheet*. https://www.analog.com/media/en/technical-documentation/data-sheets/adm7171.pdf.

[53] Elya B. Joffe and Kai-Sang Lock. *Grounds for grounding: A handbook from circuits to systems*. IEEE Press; John Wiley & Sons, Inc, 2023. ISBN: 9781119770930.

[54] NXP Semiconductors. *Hardware Development Guide for i.MX 6QuadPlus, 6Quad, 6DualPlus, Dual, 6DualLite, 6Solo Families of Applications Processors*. https://community.nxp.com/pwmxy87654/attachments/pwmxy87654/imx-processors/167183/1/IMX6DQ6SDLHDG.pdf.

[55] NXP Semiconductors. *Hardware and Layout Design Considerations for DDR Memory Interfaces*. https://www.nxp.com/docs/en/application-note/AN2582.pdf?utm_source=chatgpt.com.

[56] Advanced Micro Devices. *Zynq-7000 AP SoC and 7 Series Devices Memory Interface Solutions*. https://www.xilinx.com/support/documents/ip_documentation/mig_7series/v4_1/ug586_7Series_MIS.pdf.

[57] Analog Devices. *AD9361 Reference Manual - UG 570*. https://ez.analog.com/cfs-file/__key/telligent-evolution-components-attachments/00-441-00-00-00-07-91-97/AD9361_5F00_Reference_5F00_Manual_5F00_UG_2D00_570.pdf.

[58] Eric Bogatin. *Signal and power integrity - simplified*. Pearson Education, Limited, 2018. ISBN: 9780134513669.

[59] Madhavan Swaminathan and Ege Engin. *Power Integrity Modeling and Design for Semiconductors and Systems*. Pearson Education, Limited, 2007. ISBN: 9780132797177.

[60] Abraham I. Pressman, Keith H. Billings, and Taylor Morey. *Switching power supply design*. McGraw-Hill, 2009. ISBN: 9780071594325.

[61] YosysHQ. *PicoRV32*. https://github.com/YosysHQ/picorv32.

[62] Sarah L. Harris and David Money Harris. *Digital Design and computer architecture*. Elsevier, Morgan Kaufmann, 2016. ISBN: 9780128009116.

[63] Analog Devices. *ADAU1372 datasheet*. https://www.analog.com/media/en/technical-documentation/data-sheets/ADAU1372.pdf.

[64] *AMBA® AXITM and AceTM Protocol Specification*. URL: http://www.gstitt.ece.ufl.edu/courses/fall15/eel4720_5721/labs/refs/AXI4_specification.pdf.

[65] *AXI memory mapped to PCI Express (PCIe) Gen2*. URL: https://www.xilinx.com/products/intellectual-property/axi_pcie.html.

[66] Analog Devices. *High-Speed DMA Controller Peripheral IP*. https://wiki.analog.com/resources/fpga/docs/axi_dmac. Feb. 2023.

[67] Analog Devices. *AXI_AD9361*. https://wiki.analog.com/resources/fpga/docs/axi_ad9361. Sept. 2024.

[68] Analog Devices. *Channel CPACK Utility Core*. https://wiki.analog.com/resources/fpga/docs/util_cpack. Oct. 2021.

[69] Abrahm Silberschatz. *Operating system concepts*. Wiley, 2018. ISBN: 9781119124894.

[70] Jonathan Corbet, Alessandro Rubini, and Greg Kroah-Hartman. *Linux Device Drivers: Where the Kernel Meets the Hardware*. 3rd ed. O'Reilly Media, Feb. 2005. ISBN: 9780596555382.

[71] *Pothosware overview*. https://www.pothosware.com/.

[72] *SDRAngel*. https://www.sdrangel.org/.

[73] Maurice Herlihy and Nir Shavit. *The art of multiprocessor programming*. 2nd ed. Morgan Kaufmann, Jan. 2014. ISBN: 9780123914064.

[74] *Role of MPEG Transport Stream in Modern Broadcasting — samimgroup.com*. https://www.samimgroup.com/blog/role-mpeg-ts-broadcasting/. Feb. 2025.

[75] Gang Peng et al. "Timing and Frequency Synchronization Using CAZAC Sequences for OFDM Systems". In: *Sensors* 23.6 (2023). ISSN: 1424-8220. DOI: 10.3390/s23063168. URL: https://www.mdpi.com/1424-8220/23/6/3168.

[76] Faisal A Musa. "Noise analysis of phase locked loops and system trade-offs". In: *Department of Electrical and Computer Engineering, University of Toronto, Toronto, CA* (2002).

[77] *Gqrx SDR*. https://www.gqrx.dk/.

APPENDICES

A

APPENDIX A - IEEE MELECON 2024 PAPER

During the development of the SDR platform, presented in this thesis, a conference paper, entitled **Highly Flexible and Scalable Millimeter Wave Software Defined Radio**, was submitted, accepted, and presented at the **IEEE 22nd Mediterranean Electrotechnical Conference (MELECON)**, which took place in Porto, Portugal, on the 25th, 26th and 27th June, 2024. This paper is available on the following pages.

Highly Flexible and Scalable Millimeter Wave Software Defined Radio

João Silva*, Luís Mendes+ and João Vaz△
*+Escola Superior de Tecnologia e Gestão, Instituto Politécnico de Leiria, Leiria, Portugal
△Instituto Superior Técnico, Lisboa, Portugal
*+△Instituto de Telecomunicações, Lisboa, Portugal
E-Mail: 2222901@my.ipleiria.pt; lmendes@ipleiria.pt; joaovaz@tecnico.ulisboa.pt;

***Abstract*—This paper presents the project and implementation of a Software Defined Radio (SDR) development platform capable of operating in the Millimeter Wave (mmWave) frequency range. The SDR is composed by a base card, operating from tens of MHz up to 6 GHz, and an expansion daughter board, which performs the frequency conversion to and from mmWave bands. The base card comprises numerous programmable blocks and features, making it highly flexible by itself. The SDR can transmit and receive signals with a bandwidth of up to 56 MHz, and offloads data via Peripheral Component Interconnect - Express (PCIe) to an on-board Raspberry Pi Compute Module 4 (CM4) or, via a standardized edge connector, to any PCIe host. The platform can be powered from various sources, including Universal Serial Bus - Power Delivery (USB PD), and the power budget is over-provisioned, allowing additional expansion cards to be designed and attached to the SDR, further increasing its capabilities. This paper not only focuses on the architecture of the SDR, but also presents experimental assessment results of the most important performance characteristics.**

I. Introduction

Over the past decade, high-frequency radios with wideband capabilities, high dynamic range, spectral agility, modulation-agnostic features, and support for multiple standards have garnered significant attention from multiple fields for their diverse range of applications. Simultaneously, recent technological advancements have allowed architecture changes to radios that shrink their size, weight, power consumption, and cost, while maintaining performance and evolving toward software-programmable common hardware, that is, evolving to Software Defined Radio (SDR) devices. The operational flexibility of this type of radio allows them to be used in various fields of activity, such as telecommunications, transportation, aerospace, and defense, among other potential applications [1]. In [2], the authors utilized an SDR to characterize a communications channel in a railway environment, while in [3], an SDR was employed in a ground telecommand system for flight termination purposes.

SDRs are advanced radio systems that can be programmed to operate over a broad range of frequencies, modulations, and communication protocols. Unlike traditional radio systems, which can only be configured to operate in predetermined states, i.e., on specific frequencies and using fixed modulations schemes, SDRs utilize software-driven signal processing to adapt to different frequency bands and communication standards without requiring any hardware modifications. In addition to offering flexibility and versatility, SDRs also have several advantages over traditional radio systems, such as lower costs, higher throughput, and enhanced security. These capabilities make SDRs versatile, cost-effective, and ideal for use in modern societies where communication needs are diverse and ever-changing, such as in telecommunications and critical applications like defense, public safety and emergency response, where a flexible radio system is essential to adapt to different needs and environments.

Software Defined Radios are particularly relevant in the development of telecommunication equipment and systems, which require immense flexibility to adapt to new standards and protocols in order to meet the changing demands of users [4]. With the rapidly evolving development of wireless protocols and the increasing popularity of multi-protocol compatible devices, the SDR paradigm has proven to be very promising, especially in fifth-generation (5G) cellular networks and beyond (e.g., 6G). A Software Defined Radio enables the development and implementation of new networks, in different frequency bands, without requiring changes to hardware and allows for rapid deployment of new protocols and enhancements that enable much faster data download and upload speeds with low latency.

In the Research and Development (R&D) field, regardless of whether it is in an academic or industry environment, SDRs are becoming increasingly important due to their flexibility and agility in adapting to new communication standards and protocols. In fields such as electrical and electronic engineering, telecommunications, and computer science, SDRs are enabling researchers to design, emulate, and test a wide variety of wireless communication systems [5]. By using SDRs, researchers can rapidly prototype and evaluate communication systems in real-time, which can be especially helpful in the design phase of a project. In addition to their versatility, SDRs are also important for scientific research due to the potential for cost savings, both in terms of hardware and software. Because SDRs allow for the development of common hardware that can be programmed to fit a wide range of communication standards, researchers can potentially avoid the need for costly specialized hardware. Additionally, by taking advantage of open-source SDR software support libraries, such as [6] [7] [8] and [9], researchers can collaborate and build upon each other's work, further reducing costs and speeding up research progress.

The development and implementation of SDRs in scientific research is important because it enables researchers to stay at the forefront of rapidly evolving communication technologies. As new communication standards emerge and existing ones are updated, SDRs provide a platform for studying and developing new systems. Furthermore, by using SDRs, researchers can gain insights into how communication systems can be optimized for different applications. Overall, the continued study,

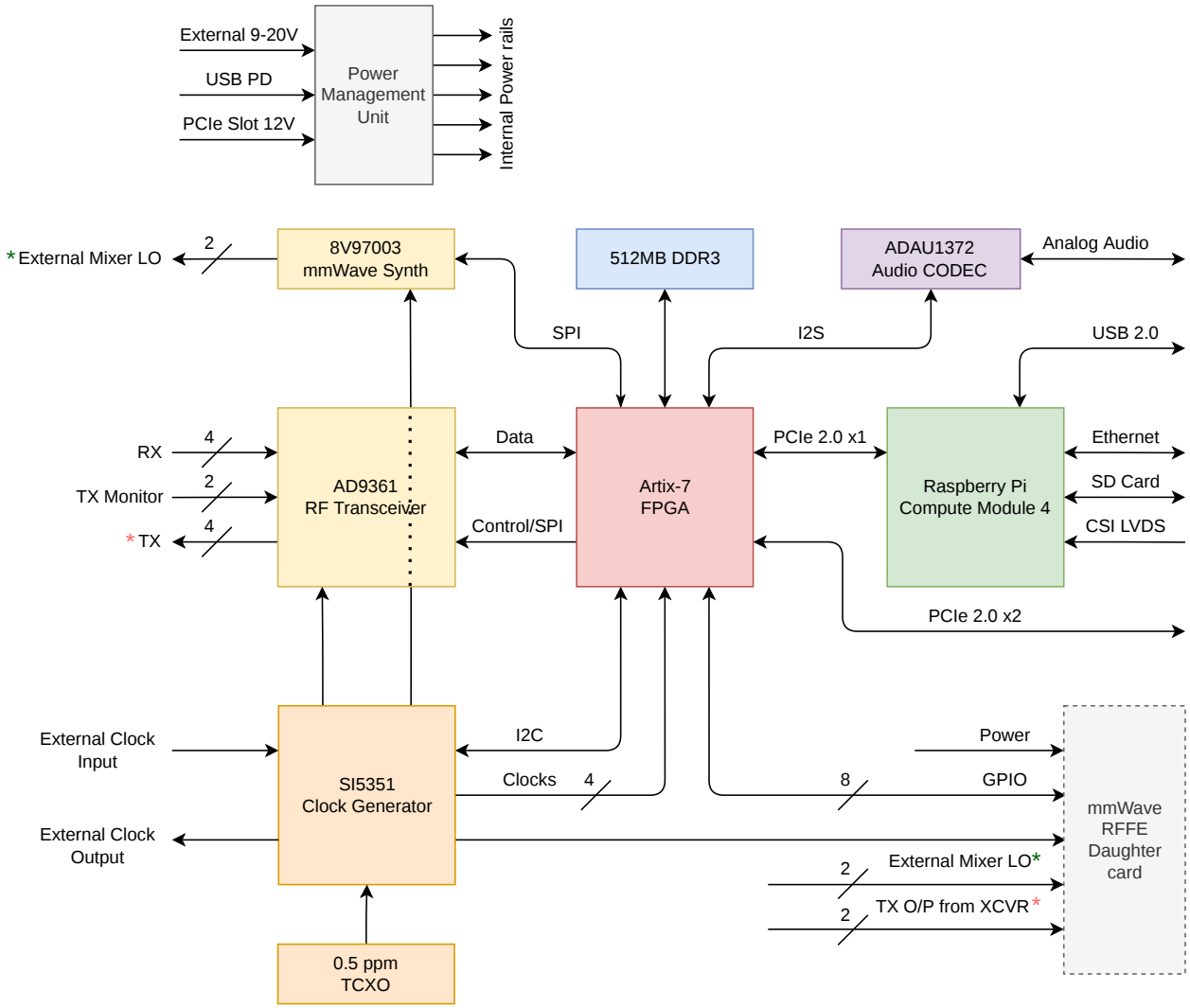


Fig. 1. SDR block diagram.

development, and implementation of SDRs in the context of scientific research is vital for staying ahead of the curve in an ever-changing field of communication technology.

As Software Defined Radios continue to evolve, they are expected to make use of higher and higher frequencies, including the Millimeter Wave (mmWave) band. While the SDR market already offers several platforms that operate at the lower end of the gigahertz frequency spectrum, the trend is moving towards higher frequencies in order to accommodate new and emerging communication technologies such as 5G [10] and complex radar systems. The mmWave bands offer a much larger bandwidth than traditional radio bands, allowing for much faster data transfer speeds with low latency. Additionally, these bands offer a large amount of unused spectrum, including unlicensed bands, which can be used for new wireless protocols and applications.

Millimeter Wave bands are well-suited for short-range communications such as indoor wireless networks, Internet of Things (IoT) devices, and Vehicle to Vehicle (V2V) communications [11]. Although they have a shorter range than traditional radio bands, this can be mitigated using beamforming and advanced signal processing techniques that are well-suited for SDRs. Another advantage of using mmWave SDRs is the high level of security that can be provided. The high frequency and directional nature of mmWave signals make them harder to intercept and jam, which can improve overall security for wireless communication systems.

Overall, the evolution of SDRs towards Millimeter Wave bands has the potential to greatly enhance wireless communication capabilities and enable new applications in a wide range of fields, such as autonomous vehicles, smart infrastructure, and the IoT. These fields require low latency, high bandwidth, and reliability.

This work presents a practical implementation of a Software Defined Radio platform capable of operating in the Millimeter Wave frequency region. Contrary to, for example, the solution proposed by [12], the architecture proposed in this paper diverges from RFSoC lineups, due to their very high cost.

This paper is organized as follows: In Section II the

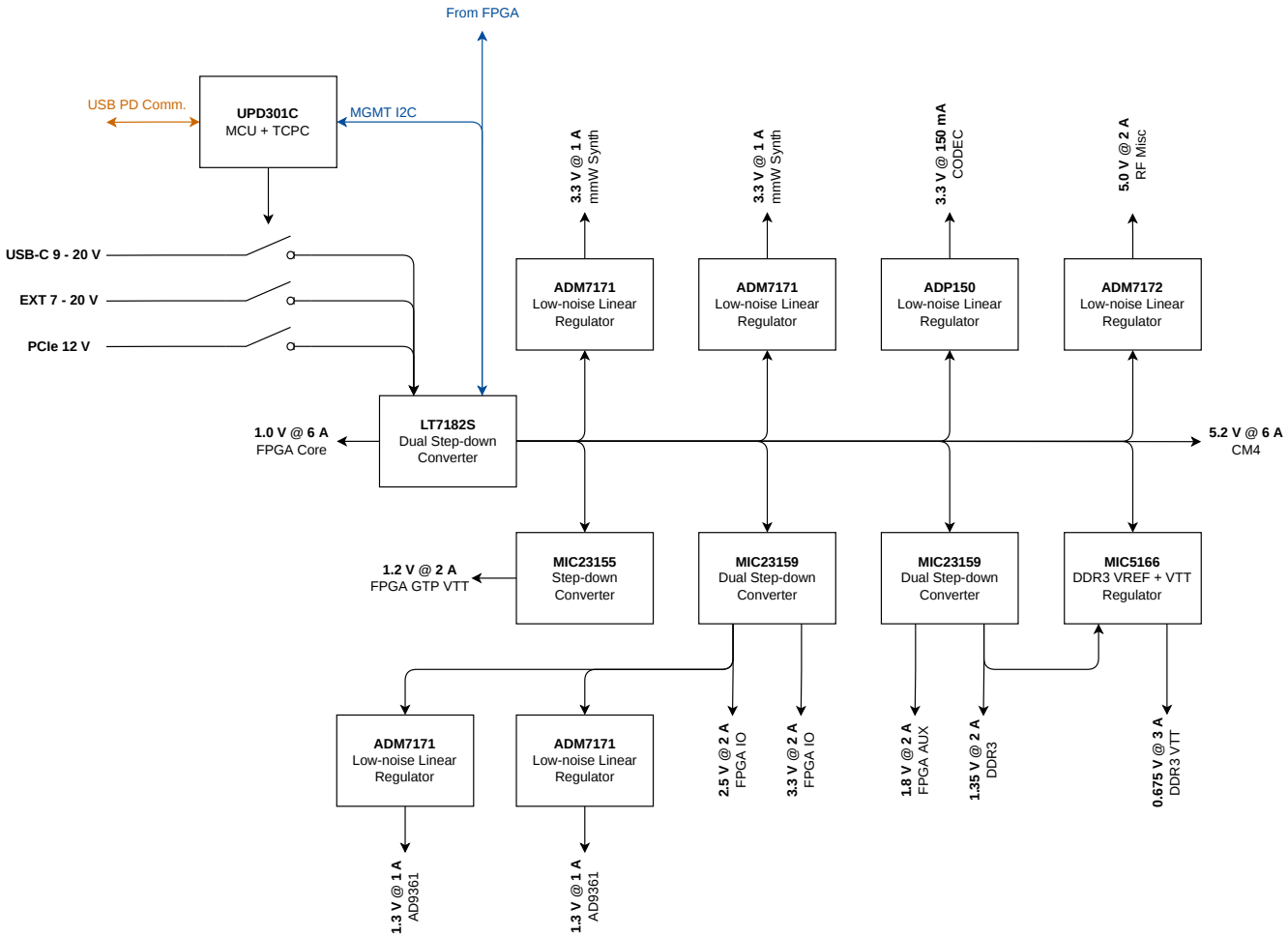


Fig. 2. SDR Power Management Unit block diagram.

overall architecture of the SDR is described. Following that, Section III briefly details the measurement procedures and presents the obtained results. Lastly, in Section IV, conclusions from the developed work are drawn.

II. SDR Architecture

This section describes, in detail, the architecture of the developed SDR. To ensure the SDR is useful for as many application scenarios as possible, it is architectured in an expansible way, leaving, where possible, room for future feature implementations. The technical specifications were also defined to cover the widest possible range of use cases.

To achieve operation in the mmWave frequency band, the proposed architecture consists of a sub-6 GHz SDR (covering frequencies ranging from tens of megahertz up to 6 GHz) along with a front-end module that extends the SDR functionality into the mmWave band, as illustrated in Fig. 1. As shown in the figure, the sub-6 GHz section comprises a Power Management Unit (PMU), a signal processing and control block, a Radiofrequency (RF) section, and a clock generation and synchronization unit.

The mmWave SDR can be powered externally through a two wire standard DC power source, via the 12 V rail of the Peripheral Component Interconnect - Express (PCIe) edge connector or via Universal Serial Bus - Power Delivery (USB PD). The PMU is responsible for generating, controlling, sequencing and monitoring all the power supply rails required by the internal functional blocks of the radio. It comprises a Power Management Controller (PMC) microcontroller, which, among other tasks, handles USB PD communication and negotiation, selects the best power source to use and correctly configures the voltage regulators to produce all the voltage rails required by the system, while also sequencing them properly. Fig. 2 depicts the SDR power supply tree, including the three aforementioned possible power inputs, all the voltage regulators, voltage levels and maximum current supported by each rail. This figure also contains the PMC and its connections to the power tree.

The synchronization unit generates clocks for the digital logic and reference signals for various functional blocks (i.e., transceiver and frequency synthesizers), assuring the required synchronization between clock domains. It is comprised by a Temperature-Compensated Crystal Oscillator (TCXO) and a highly programmable clock generator Integrated Circuit (IC). The synchronization unit allows the SDR user to input an external clock source (e.g., from a GPS Disciplined Oscillator

(GPSDO)) as well as providing the user with a clock output for synchronizing external devices. The highly flexible clock generator Integrated Circuit (IC) is fully programmable, allowing the user to select which clock source to use (internal TCXO or external clock) for a certain clock output, as well as setting the division and multiplication ratio, in order to achieve a determined output frequency.

The signal processing and SDR control block is where all the digital signal processing occurs, such as modulation, demodulation, filtering and block configuration, control, and monitoring. This block serves as the "brain" of the radio and comprises a Field-Programmable Gate Array (FPGA), which communicates via PCIe with an on-board Raspberry Pi Compute Module 4 (CM4) or with an external Root Complex (host) through the edge connector. The FPGA has more than enough logic resources for accelerating and even completely implementing Digital Signal Processing (DSP) algorithms, although it can be programmed to serve simply as a translation layer between the RF transceiver and the host controller. 512 MB of Double Data Rate 3 (DDR3) Random Access Memory (RAM) are also included and directly connected to the FPGA, which further increase the flexibility of the SDR when operating standalone.

The RF section comprises an agile RF transceiver which performs digital filtering, signal domain conversion, frequency conversion (translation) and signal amplification. The agile transceiver operates up to 6 GHz, but its output signals can be routed to the mmWave module which performs the translation to and from the mmWave band. The base platform also includes a frequency synthesizer IC capable of generating signals with frequency up to 18 GHz. These are meant to serve as Local Oscillator (LO) for the mmWave module.

The base platform is designed such that further expansion is possible. In the future, additional expansion cards other than the mmWave module can be developed, either to extend the frequency span to higher or lower frequencies, as well as providing higher output powers, among other suitable features. The expansion modules have access to the RF transceiver inputs and outputs, the frequency synthesizer outputs, various power rails with different voltages and current supply capabilities, general purpose digital I/O to/from the FPGA and a clock signal from the synchronization unit. Fig. 3 represents an example board layout of an expansion module for the SDR, including all the available connections, as mentioned previously. This debug expansion module was produced for testing purposes, while the mmWave card is still in the development phase.

Although not an essential component for the SDR operation, an audio Coder / Decoder (CODEC) is also included in the design, as part of the flexible architecture objective. This IC allows easier interfacing with analog audio peripherals.

Fig. 4 depicts the fully assembled SDR Printed Circuit Board (PCB), which consists in a 6-layer FR-4 stackup. The main constituent blocks present in Fig. 1 are also shown in this figure for reference.

III. Implementation and Experimental Assessment

Preliminary assessment of the base SDR platform transmitter (up to 6 GHz) was performed by configuring the transmitter

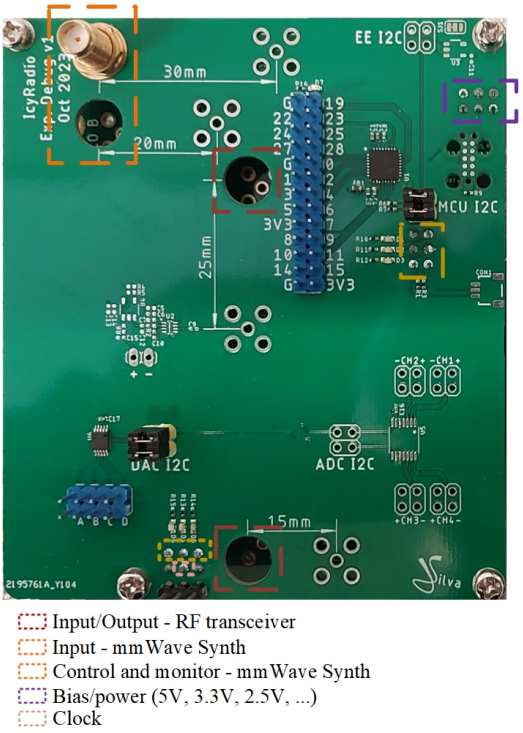


Fig. 3. SDR test expansion card/module PCB.

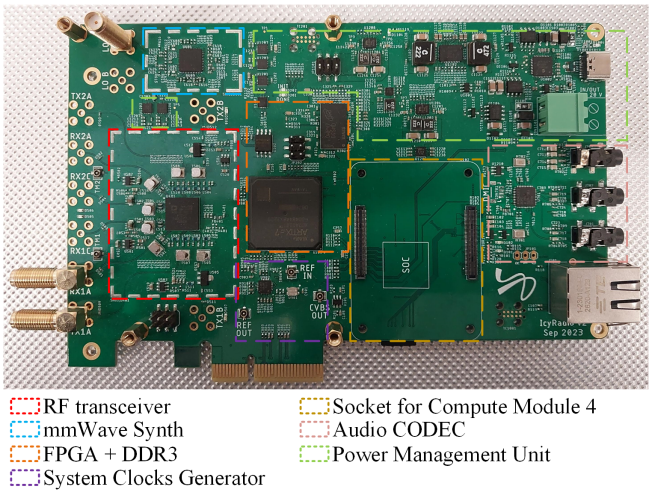


Fig. 4. SDR assembled PCB.

to output a carrier modulated by band-limited white noise at various central frequencies. The FPGA was used to feed the RF transceiver with random data, and the transceiver's build-in configurable Finite Impulse Response (FIR) filter was used to limit the signal bandwidth to approximately 1.1 MHz.

One of the base SDR's transmitter outputs was directly connected to a *Rohde & Schwarz* FPL1007 spectrum analyzer, configured for Adjacent Channel Power Ratio (ACPR) measurement. Table I summarizes the obtained ACPR results, measured at different center frequencies, at the transceiver's maximum output power (internal transmitter attenuation control configured for 0 dB). As can be seen, the ACPR is worse

TABLE I. ACPR RESULT SUMMARY AT MAXIMUM POWER.

100 MHz	500 MHz	1 GHz	3 GHz	6 GHz
-40.23 dBc	-49.90 dBc	-45.29 dBc	-38.77 dBc	-33.03 dBc

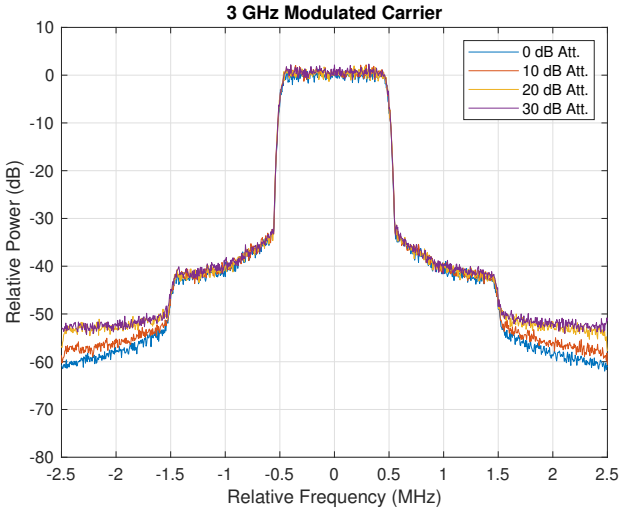


Fig. 5. Measured transmitter output spectrum at 3 GHz.

at the limit frequencies (100 MHz and 6 GHz) and attains its best value of approximately -50 dBc at 500 MHz. However, even at 6 GHz, where the ACPR achieves its worst value, the transmit path of the SDR can be considered as having a good linearity.

The measured output signal spectra obtained at 3 GHz for four different transmitter path attenuations are illustrated in Fig. 5. The presented signals spectra are all normalized by the power at the center of the band. As can be seen, the used attenuations do not have a visible impact on the close-in out-of-band spectrum, since the sidebands are mainly originated in the digital signal processing blocks. In this figure, the effect of the sampling rate on the signal spectrum is also visible, i.e., the abrupt power decrease at ± 1.5 MHz from the carrier. Fig. 6 shows the spectrum of the five modulated carriers, 10 dB below the maximum power level. The figure shows that the output power increases when the frequency increases until it reaches 1 GHz. From there, the output power decreases monotonically.

Spurious and harmonic emissions were checked by connecting one of the transmitter outputs to an *Agilent* E4407B spectrum analyzer while transmitting a single sinusoidal tone generated by a Digital Direct Synthesizer (DDS) instantiated inside the FPGA. Fig. 7 depicts the used lab bench setup, where the sub-6 GHz SDR is loaded with the CM4 and the placeholder daughter board (where the mmWave front-end will be installed). In this configuration, the SDR works standalone, i.e., the configuration, control and processing are all internal (on the board) and only the monitoring is remotely performed through cabled network. Fig. 8 shows the spectral footprint of the SDR, from DC up to 26.5 GHz, while transmitting a single carrier at 3 GHz, at the maximum output power. It is possible to observe that the resulting spectrum is quite clean, given that only the fundamental tone and its harmonics show up above the analyzer noise floor. Table II summarizes the amplitudes of the visible tones.

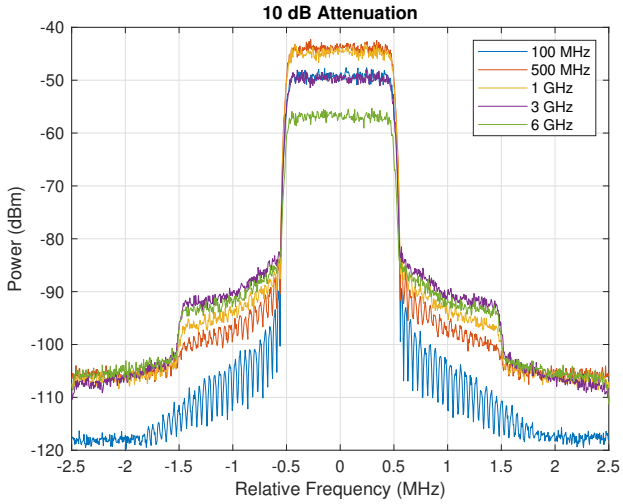


Fig. 6. Measured transmitter output spectra at various frequencies, 10 dB below the maximum output power.

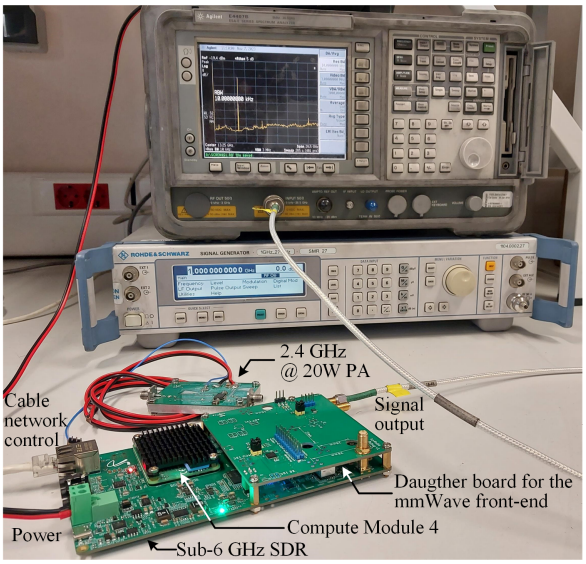


Fig. 7. Bench setup for wideband spectrum measurement.

TABLE II. WIDEBAND SPECTRUM TONE POWER.

3 GHz (Fundam.)	6 GHz	9 GHz	12 GHz	15 GHz
-21.85 dBm	-75.46 dBm	-71.70 dBm	-65.15 dBm	-68.88 dBm

The receiver path of the sub-6 GHz SDR was also assessed. For that, one of the RF inputs was fed with carriers signals at different frequencies, generated by a *Rohde & Schwarz* SMR 27 signal generator. The SDR was also tested as an spectrum analyzer and an FM radio station receiver. All these experimental tests run well, showing that the receiver path of the sub-6 GHz SDR also works as expected.

The performance of the mmWave frequency synthesizer was also experimentally evaluated, namely, its phase noise profile was obtained utilizing an *Agilent* E4407B spectrum analyzer configured for phase noise analysis. Fig. 9 shows the results obtained while the synthesizer was locked at 18 GHz. Its output power was backed-off from the maximum

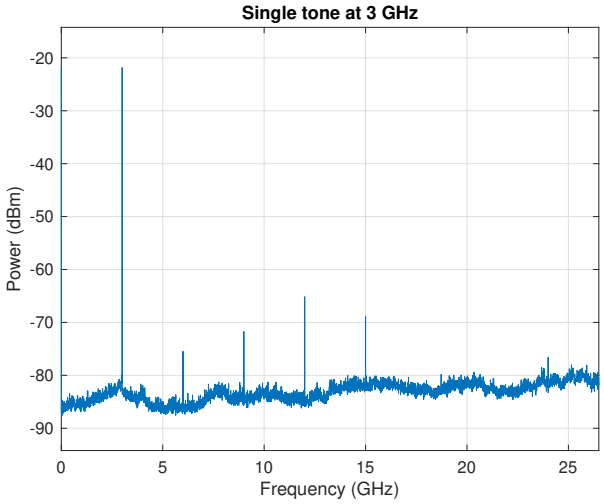


Fig. 8. Measured SDR wideband output spectral footprint.

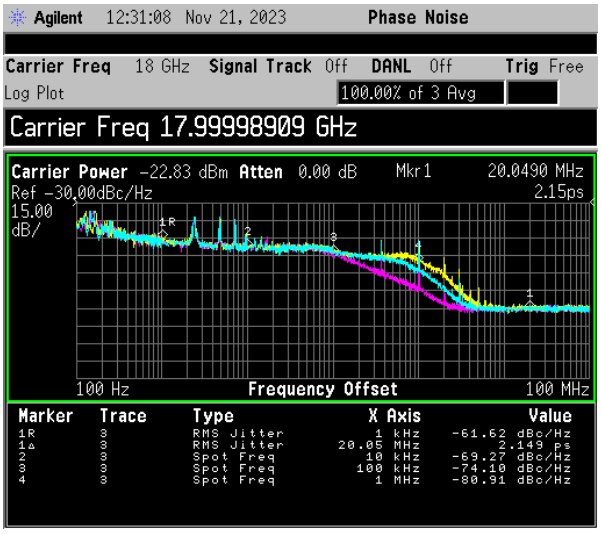


Fig. 9. Measured SDR mmWave Synthesizer phase noise at 18 GHz.

value to prevent possible output stage compression artifacts from contaminating the results. In this figure, three curves are shown, for different charge pump current values, whose variations directly affect the bandwidth of the loop filter. The yellow curve represents the lowest charge pump current setting, and, consequently, the widest bandwidth filter. The blue curve represents the mid-scale charge pump current setting, and the magenta curve the maximum value, yielding the narrowest filter bandwidth. The effect of the charge pump current on the filter bandwidth, and, consequently, on the phase noise profile is clearly seen.

The presented phase noise results were obtaining by using a wide loop filter bandwidth, in order to minimize the time required to achieve a proper frequency and phase lock, and also such that the phase noise of the synthesizer was dominated by the reference phase noise. In the current design, the synthesizer is not intended for fast frequency hopping applications, and is, instead, going to be used to provide a Local Oscillator signal to expansion modules, which should only require periodic and quite coarse frequency jumps. Given this fact, a much narrower loop bandwidth is tolerated, since fast lock times are not critical, and narrowing the loop bandwidth will certainly alleviate, to some extent, the shortcomings in the phase noise of the synthesized tones.

IV. Conclusion

In the past decade, high-frequency radios with advanced capabilities have gained attention across various fields, leading to the evolution of SDR devices. Particularly relevant in telecommunications, aerospace, defense and R&D, SDRs play a crucial role in adapting to evolving wireless protocols. The SDR presented in this paper is highly flexible, programmable and adaptable thanks to the utilized components (FPGA, CM4, etc...) and the multiple ways in which they can be interconnected and operated. The paper also presents real (measured) performance results regarding the base SDR board (up to 6 GHz), which traduce good transmitter RF performance. Experimental results are also provided for the mmWave frequency synthesizer, which achieves the expected performance levels. Development is still ongoing, especially regarding the mmWave expansion card, and further testing will be required to fully evaluate the SDR RF performance.

Acknowledgement

The authors would like to thank the Instituto de Telecomunicações, Wireless Circuits - LX research group, for their support.

References

[1] T. Nesimoglu, “A review of software defined radio enabling technologies,” in *2010 10th Mediterranean Microwave Symposium*, 2010, pp. 87–90.

[2] Y. Liang, H. Li, Y. Tian, Y. Li, and W. Wang, “Sdr-based 28 ghz mmwave channel modeling of railway marshaling yard,” *Sensors*, vol. 23, no. 19, 2023. [Online]. Available: https://www.mdpi.com/1424-8220/23/19/8108

[3] M. K. Gourab, A. Roy, and U. Mandal, “Design of ground telecommand system for flight termination in a software defined radio platform,” in *2023 3rd International Conference on Range Technology (ICORT)*, 2023, pp. 1–6.

[4] T. Natarajan and C. Kh, “A review on recent trends in software defined radio design and applications,” *International Journal of Advanced Research in Electronics and Communication Engineering (IJARECE)*, vol. 6, pp. 1021–1025, 10 2017.

[5] R. Akeela and B. Dezfouli, “Software-defined radios: Architecture, state-of-the-art, and challenges,” *Computer Communications*, vol. 128, pp. 106–125, 2018. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0140366418302937

[6] A. Retzler, “csdr,” https://github.com/ha7ilm/csdr.

[7] “Gnu radio,” https://www.gnuradio.org/.

[8] J. D. Gaeddert, “liquid-dsp,” https://github.com/jgaeddert/liquid-dsp.

[9] Pothosware, “Soapysdr,” https://github.com/pothosware/SoapySDR.

[10] Y. Fujii, T. Iye, K. Tsuda, and A. Tanibayashi, “28ghz cooperative digital beamforming for 5g advanced system on an sdr platform,” in *2021 IEEE Radio and Wireless Symposium (RWS)*, 2021, pp. 80–82.

[11] K. Singh, P. Rawat, and J.-M. Bonnin, “Cognitive radio for vehicular ad hoc networks (cr-vanets): Approaches and challenges,” *EURASIP Journal on Wireless Communications and Networking*, vol. 2014, p. 49, 03 2014.

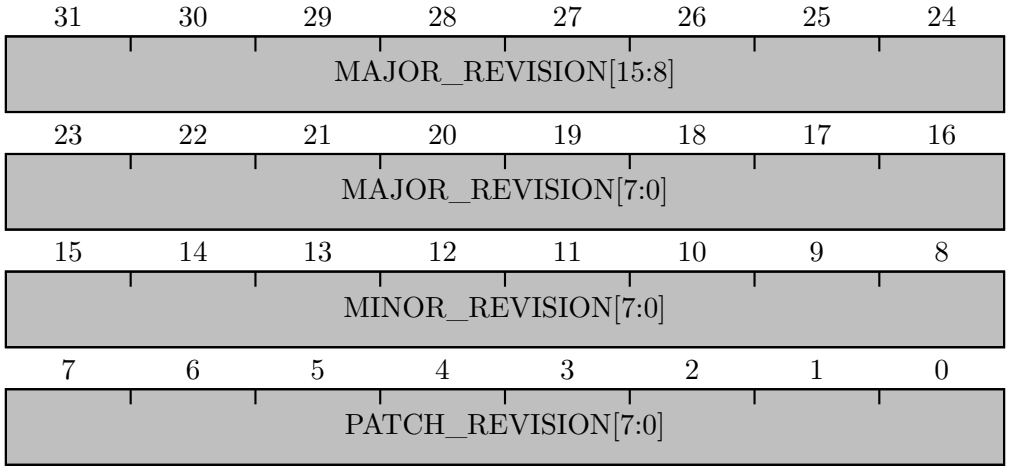
[12] A. Şahin, M. L. Sichitiu, and I. Guvenç, “A millimeter-wave software-defined radio for wireless experimentation,” in *IEEE INFOCOM 2023 - IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, 2023, pp. 1–6.

B

APPENDIX B - CUSTOM IP CORES REGISTER MAP

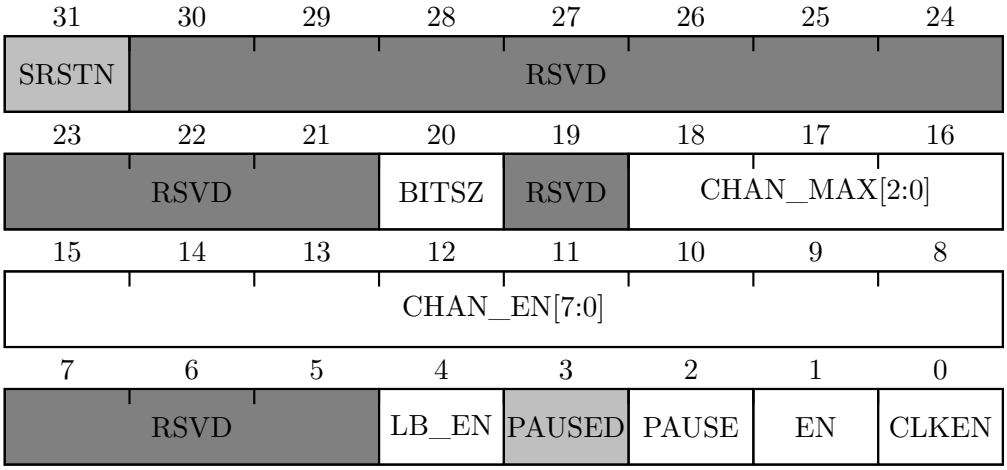
B.1 AXI I2S ENGINE IP REGISTER DOCUMENTATION

B.1.1 *Register 0 - 0x00 - IP Version*



Bitfield	Access	Default value	Description
MAJOR_REVISION[15:0]	R	'h0001	IP module major revision v[**MAJ**].[MIN].[PATCH]
MINOR_REVISION[7:0]	R	'h00	IP module minor revision v[MAJ].[**MIN**].[PATCH]
PATCH_REVISION[7:0]	R	'h00	IP module patch revision v[MAJ].[MIN].[**PATCH**]

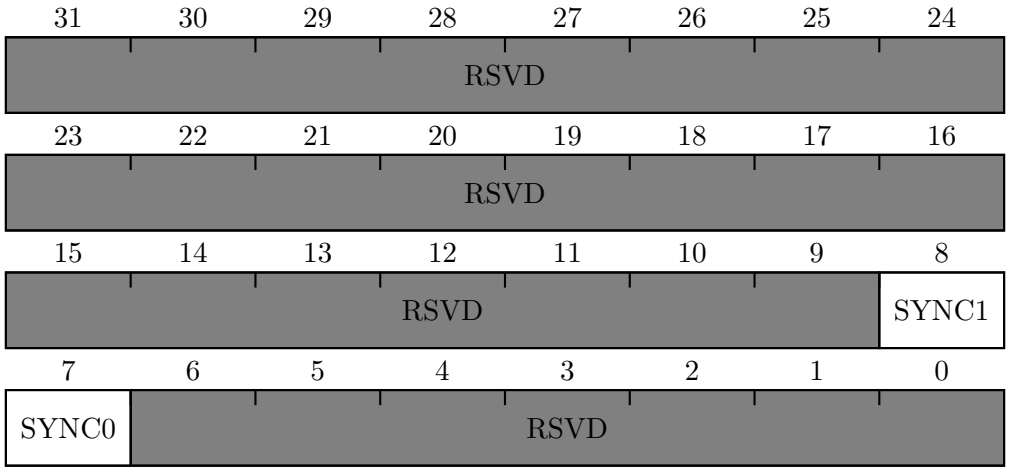
B.1.2 *Register 1 - 0x04 - Status and Control*



Bitfield	Access	Default value	Description
SRSTN	R	'b0	I2S and AXI-Stream clock domain synchronous reset (inverted), synchronized to the AXI-Lite clock domain. Used for debug purposes.
BITSZ	R/W	'b0	Channel bit size ('b0 = 16 bits, 'b1 = 32 bits). Set to 'b1 for 24-bit size (packed into 32-bit word). **This field can only be changed while the I2S is disabled (EN = 'b0).**
CHAN_MAX[2:0]	R/W	'b000	Number of channels in the frame minus one. **This field can only be changed while the I2S is disabled (EN = 'b0).**
CHAN_EN[7:0]	R/W	'h00	Bit mask of enabled slots (channels). Each bit corresponds to a channel. **This field can only be changed while the I2S is disabled (EN = 'b0).**
LB_EN	R/W	'b0	Enable loopback mode.
PAUSED	R	'b0	Whether the I2S FSM is paused.
PAUSE	R/W	'b0	Request the I2S FSM to be paused.

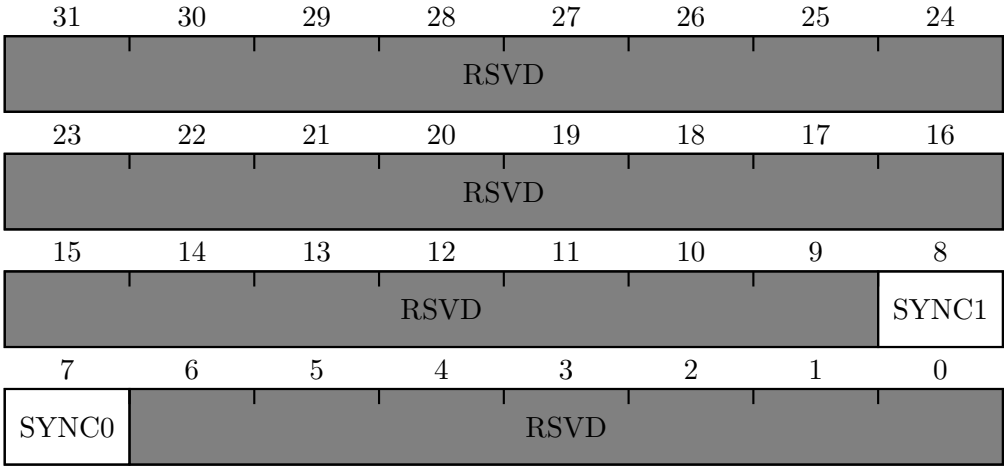
Bitfield	Access	Default value	Description
EN	R/W	'b0	Enable the I2S FSM. **The FSM can only be enabled when the clocks are enabled (CLKEN = 'b1).**
CLKEN	R/W	'b0	Enable the I2S clock divider. **The clock divider can only be disabled when the I2S FSM is disabled (EN = 'b0).**

B.1.3 *Register 2 - 0x08 - IRQ Enable*



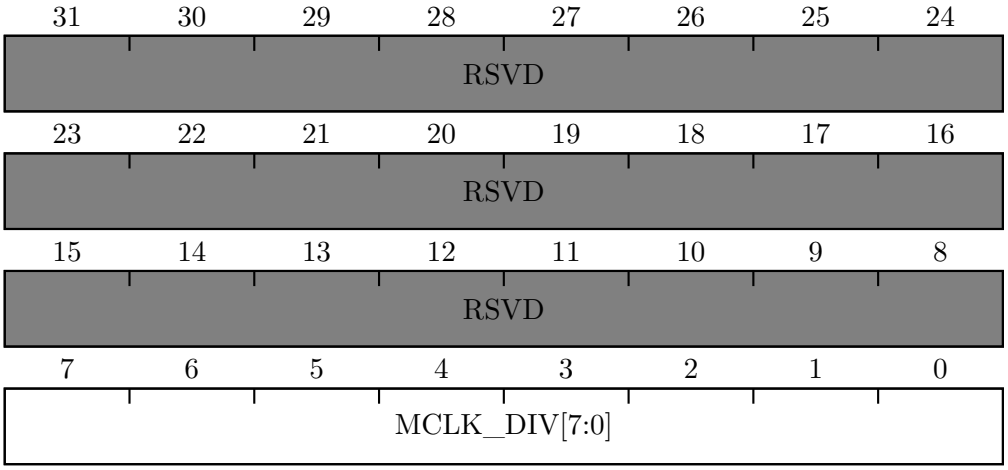
Bitfield	Access	Default value	Description
SYNC1	R/W	'b0	IRQ mask for CDC synchronization handshake completion from *i2s_src_clk* to *aclk* ('b0 = Masked, 'b1 = Unmasked). Used for debug purposes.
SYNC0	R/W	'b0	IRQ mask for CDC synchronization handshake completion from *aclk* to *i2s_src_clk* ('b0 = Masked, 'b1 = Unmasked). Used for debug purposes.

B.1.4 *Register 3 - 0x0C - IRQ Pending*



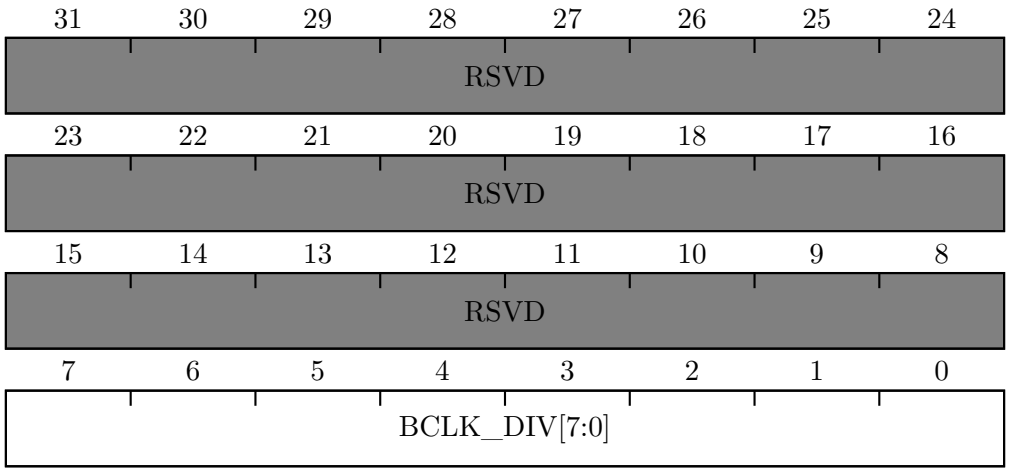
Bitfield	Access	Default value	Description
SYNC1	R/W1TC	'b0	CDC synchronization handshake from *i2s_src_clk* to *aclk* is complete. Write 1 to clear the pending interrupt flag. Used for debug purposes.
SYNC0	R/W1TC	'b0	CDC synchronization handshake from *aclk* to *i2s_src_clk* is complete. Write 1 to clear the pending interrupt flag. Used for debug purposes.

B.1.5 *Register 4 - 0x10 - MCLK Divider Ratio*



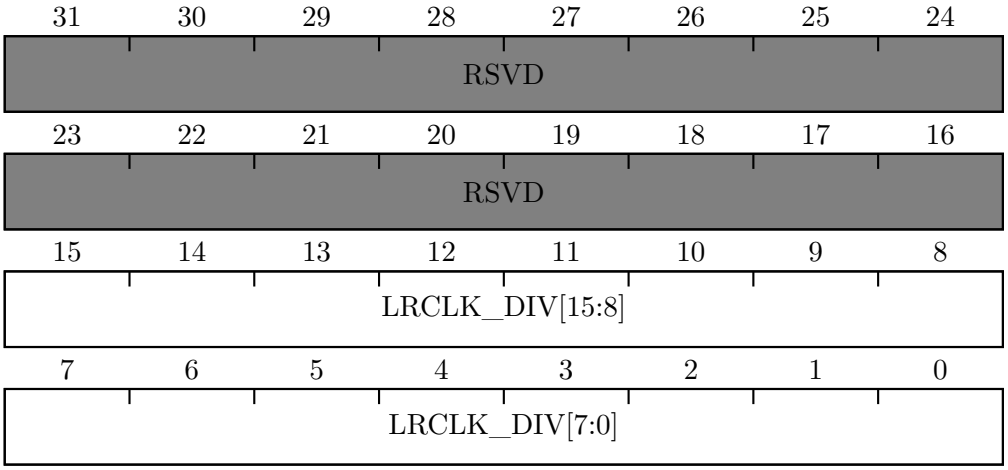
Bitfield	Access	Default value	Description
MCLK_DIV[7:0]	R	'h00	I2S MCLK clock divider ratio. This value determines the relationship between the I2S input clock (f_{src}) and the MCLK clock (f_{MCLK}). $f_{MCLK} = \frac{f_{src}}{2 \cdot (MCLK\_DIV[7:0]+1)}$ **This field can only be changed while the I2S clock divider is disabled (CLKEN = 'b0).**

B.1.6 *Register 5 - 0x14 - BCLK Divider Ratio*



Bitfield	Access	Default value	Description
BCLK_DIV[7:0]	R	'h00	I2S BCLK clock divider ratio. This value determines the relationship between the I2S input clock (f_{src}) and the BCLK clock (f_{BCLK}). $f_{BCLK} = \frac{f_{src}}{2 \cdot (BCLK\_DIV[7:0]+1)}$ **This field can only be changed while the I2S clock divider is disabled (CLKEN = 'b0).**

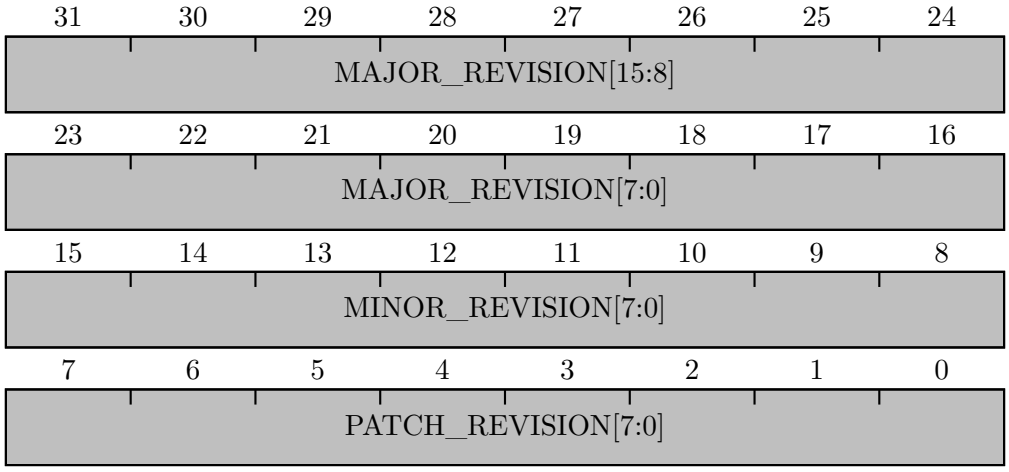
B.1.7 *Register 6 - 0x18 - LRCLK Divider Ratio*



Bitfield	Access	Default value	Description
LRCLK_DIV[15:0]	R	'h00	I2S LRCLK clock divider ratio. This value determines the relationship between the I2S input clock (f_{src}) and the LRCLK clock (f_{LRCLK}). $f_{LRCLK} = \frac{f_{src}}{2 \cdot (LRCLK\_DIV[7:0]+1)}$ **This field can only be changed while the I2S clock divider is disabled (CLKEN = 'b0).**

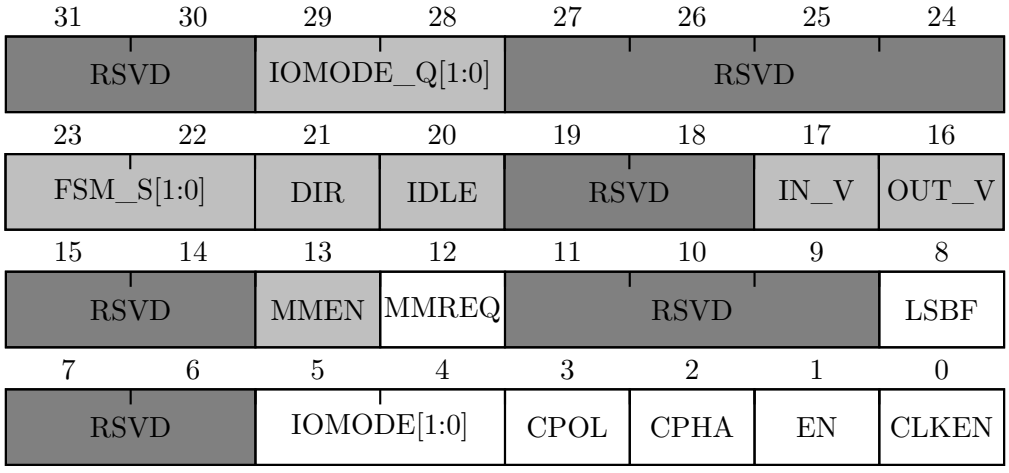
B.2 AXI SPI ENGINE IP REGISTER DOCUMENTATION

B.2.1 *Register 0 - 0x00 - IP Version*



Bitfield	Access	Default value	Description
MAJOR_REVISION[15:0]	R	'h0001	IP module major revision v[**MAJ**].[MIN].[PATCH]
MINOR_REVISION[7:0]	R	'h00	IP module minor revision v[MAJ].[**MIN**].[PATCH]
PATCH_REVISION[7:0]	R	'h00	IP module patch revision v[MAJ].[MIN].[**PATCH**]

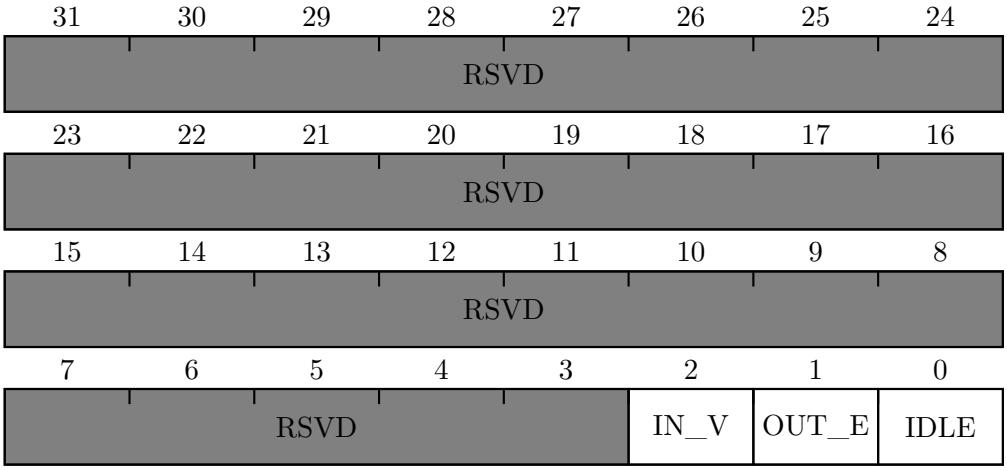
B.2.2 *Register 1 - 0x04 - Status and Control*



Bitfield	Access	Default value	Description
IOMODE_Q[1:0]	R	'b00	Current IO mode latched by the SPI FSM. Used for debug purposes.
FSMS[1:0]	R	'b00	Current SPI FSM state. Used for debug purposes.
DIR	R	'b0	Current transaction direction latched by the SPI FSM. Used for debug purposes.
IDLE	R	'b0	Whether the SPI FSM is idle or not.
IN_V	R	'b0	Whether the SPI input data buffer contains valid data or not.
OUT_V	R	'b0	Whether the SPI output data buffer contains valid data or not.
MMEN	R	'b0	Whether the SPI memory-mapped IO mode is enabled. **This bit is only available if the core is configured to support MMIO mode, otherwise it's reserved.**
MMREQ	R/W	'b0	Request to enable ('b1) or disable ('b0) the SPI memory-mapped IO mode. The actual status of this mode is available via the MMEN bit. **This bit is only available if the core is configured to support MMIO mode, otherwise it's reserved.**
LSBF	R/W	'b0	Select between LSB-first mode ('b1) or MSB-first mode ('b0). **This field can only be changed while the SPI is idle (IDLE = 'b1) and MMIO mode is not present or disabled (MMEN = 'b0).**

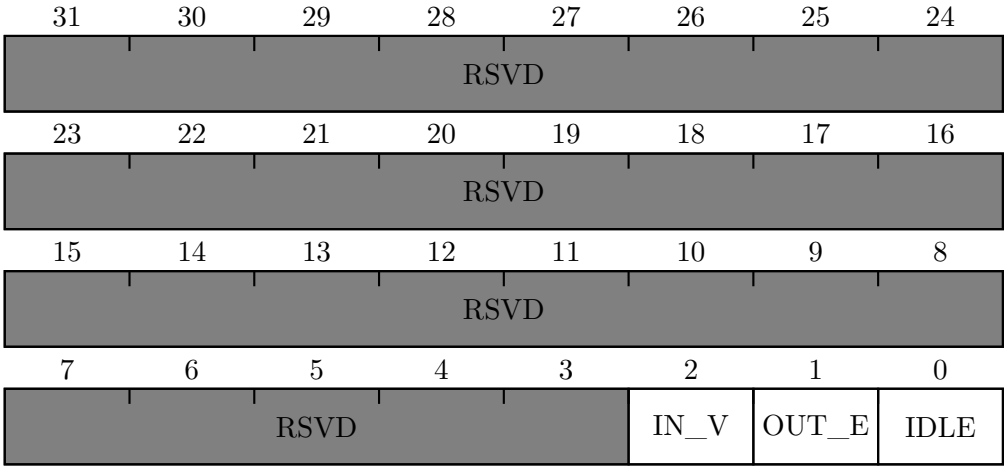
Bitfield	Access	Default value	Description
IOMODE[1:0]	R/W	'b00	Select the SPI IO mode to use on the next transaction: **'b00**: Standard SPI. MOSI is always an output and MISO is always an input, independently of the transaction direction. **'b01**: 3-wire mode. MISO is not used. MOSI IO direction is dependent on the transaction direction. **'b10**: Dual IO mode. MOSI is D0 and MISO is D1. IO direction is dependent on the transaction direction and two bits are clocked in/out at every SCK cycle. **'b11**: Quad IO mode. MOSI is D0, MISO is D1, D2 and D3 are also used. IO direction is dependent on the transaction direction and four bits are clocked in/out at every SCK cycle. **This mode is only available if the core is configured to support quad mode. Otherwise this setting will not apply and the previous value will be kept.**
CPOL	R/W	'b0	Select the SCK idle value. This bit is the MSB of the SPI mode. **This field can only be changed while the SPI is idle (IDLE = 'b1).**
CPHA	R/W	'b0	Select the SCK sample/setup edge. This bit is the LSB of the SPI mode. **This field can only be changed while the SPI is idle (IDLE = 'b1).**
EN	R/W	'b0	Enable the SPI FSM. **The FSM can only be enabled when the clocks are enabled (CLKEN = 'b1).**
CLKEN	R/W	'b0	Enable the SPI clock divider. **The clock divider can only be disabled when the SPI FSM is disabled (EN = 'b0).**

B.2.3 *Register 2 - 0x08 - IRQ Enable*



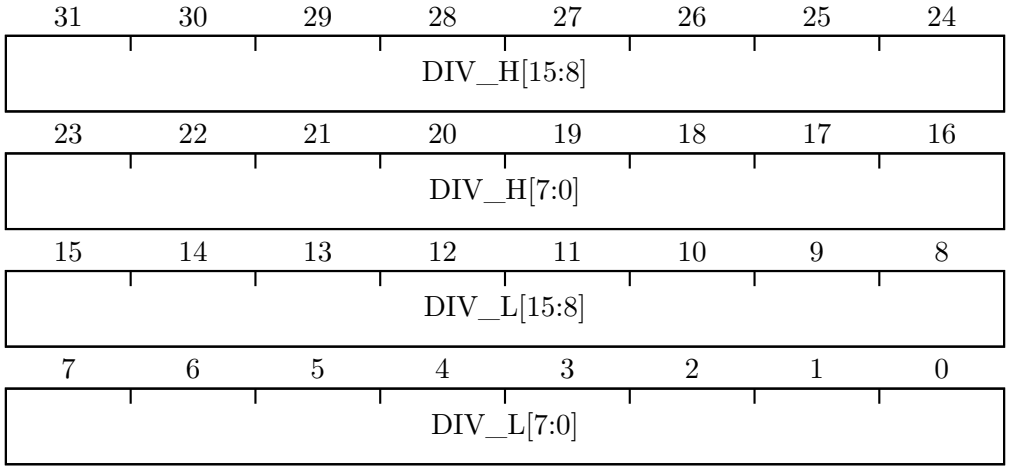
Bitfield	Access	Default value	Description
IN_V	R/W	'b0	IRQ mask for when valid data becomes available on the SPI input data buffer ('b0 = Masked, 'b1 = Unmasked).
OUT_E	R/W	'b0	IRQ mask for when space becomes available on the SPI output data buffer ('b0 = Masked, 'b1 = Unmasked).
IDLE	R/W	'b0	IRQ mask for when the SPI FSM becomes idle ('b0 = Masked, 'b1 = Unmasked).

B.2.4 *Register 3 - 0x0C - IRQ Pending*



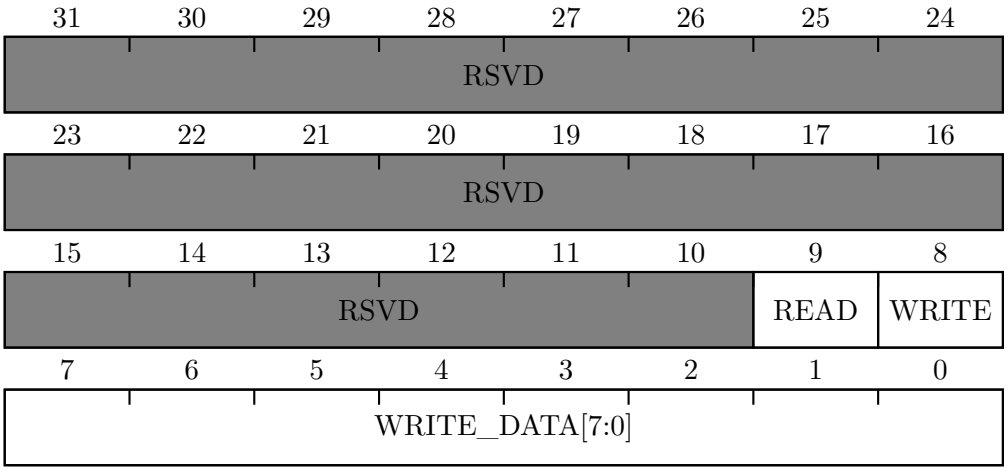
Bitfield	Access	Default value	Description
IN_V	R/W1TC	'b0	Valid data is available on the SPI input data buffer. Write 1 to clear the pending interrupt flag.
OUT_E	R/W1TC	'b0	Space is available on the SPI output data buffer. Write 1 to clear the pending interrupt flag.
IDLE	R/W1TC	'b0	SPI FSM is idle. Write 1 to clear the pending interrupt flag.

B.2.5 *Register 4 - 0x10 - SCK Divider Ratio*



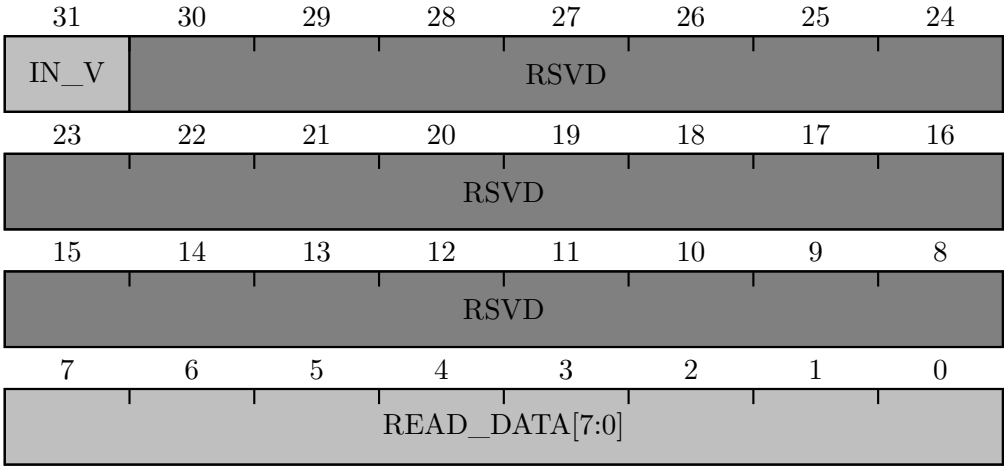
Bitfield	Access	Default value	Description
DIV_H[15:0]	R	'h0000	SPI SCK divider counter top value when the clock is high. This value, together with DIV_L[15:0] determines the relationship between the input clock (f_{aclk}) and the SCK clock (f_{SCK}). $f_{SCK} = \frac{f_{aclk}}{DIV\_H[15:0]+DIV\_L[15:0]+2}$ The f_{SCK} duty cycle, D_{SCK}, when CPOL = 'b0, can be determined by: $D_{SCK} = \frac{DIV\_H[15:0]+1}{DIV\_H[15:0]+DIV\_L[15:0]+2}$. When CPOL = 'b1, the duty cycle is given by: $D_{SCK} = \frac{DIV\_L[15:0]+1}{DIV\_H[15:0]+DIV\_L[15:0]+2}$. **This field can only be changed while the SPI clock divider is disabled (CLKEN = 'b0).**
DIV_L[15:0]	R	'h0000	SPI SCK divider counter top value when the clock is low. This value, together with DIV_H[15:0] determines the relationship between the input clock (f_{aclk}) and the SCK clock (f_{SCK}). **This field can only be changed while the SPI clock divider is disabled (CLKEN = 'b0).**

B.2.6 *Register 5 - 0x14 - Command and output buffer*



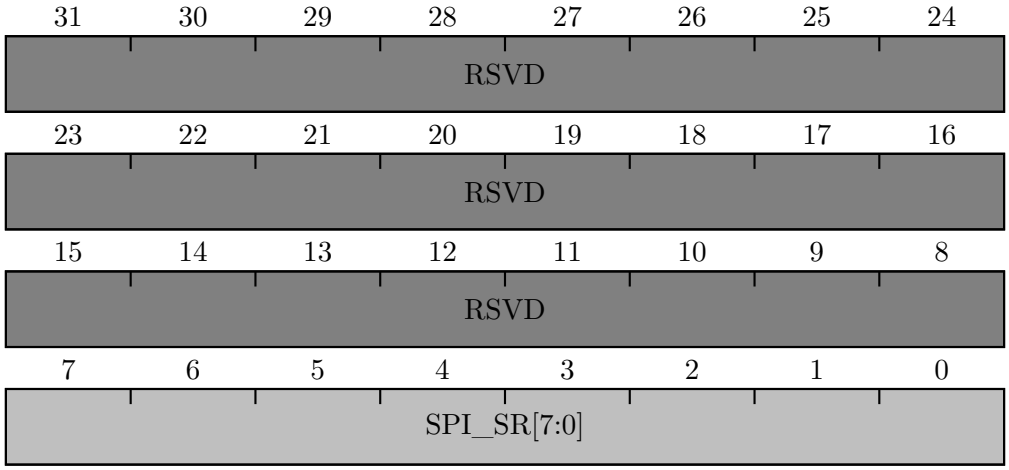
Bitfield	Access	Default value	Description
READ	R/W	'b0	Write 'b1 to this bit to place a read request to the SPI FSM. This bit will read 'b1 while the request is pending and will de-assert to 'b0 when the request has been accepted by the FSM. Another request can then be placed, even though the previous one may not be complete already. The WRITE bit has a higher priority than the READ bit. If both are set, the READ bit is ignored.
WRITE	R/W	'b0	Write 'b1 to this bit to place a write request to the SPI FSM. WRITE_DATA[7:0] should also be set to the desired data to be written in the same access. This bit will read 'b1 while the request is pending and will de-assert to 'b0 when the request has been accepted by the FSM. Another request can then be placed, even though the previous one may not be complete already. The WRITE bit has a higher priority than the READ bit. If both are set, the READ bit is ignored.
WRITE_DATA[7:0]	R/W	'h00	Data byte to be written when the WRITE bit is set, ignored otherwise. Read data corresponds to the data associated with the pending write command, if WRITE is also set.

B.2.7 *Register 6 - 0x18 - Input buffer*



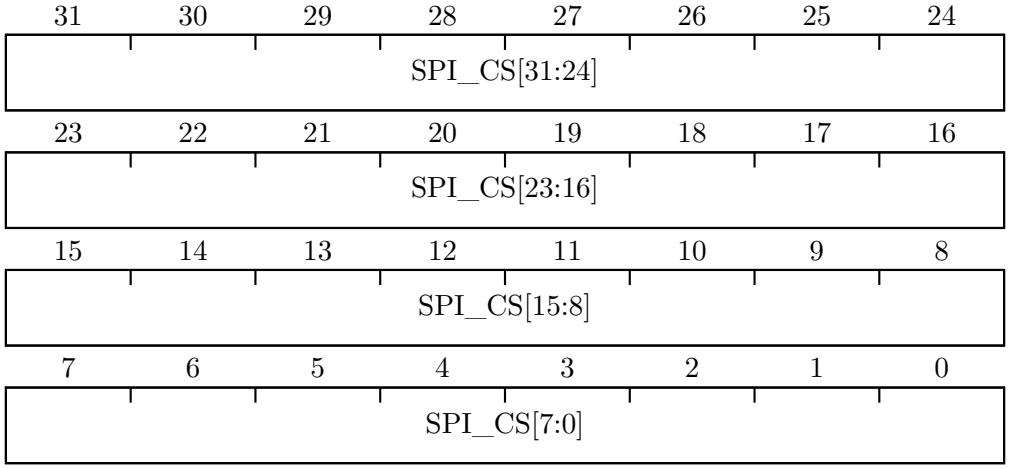
Bitfield	Access	Default value	Description
IN_V	R	'b0	This bit is an alias of the IN_V bit of the Control and Status Register (appendix B.2.2).
READ_DATA[7:0]	R	'h00	SPI input data buffer. When IN_V is 'b0, this data is undefined. When IN_V is 'b1, this field contains valid data, and the buffer is cleared afterwards. **Accesses to this register should always be 32-bit wide, as the IN_V bit is necessary to determine the validity of READ_DATA[7:0]. When MMIO mode is supported and enabled (MMEN = 'b1), reads to this register do not clear the buffer.**

B.2.8 *Register 7 - 0x1C - Shift-register peek*



Bitfield	Access	Default value	Description
SPI_SR[7:0]	R	'h00	Internal SPI shift register. Read operations to this register do not cause any side effects. Used for debug purposes.

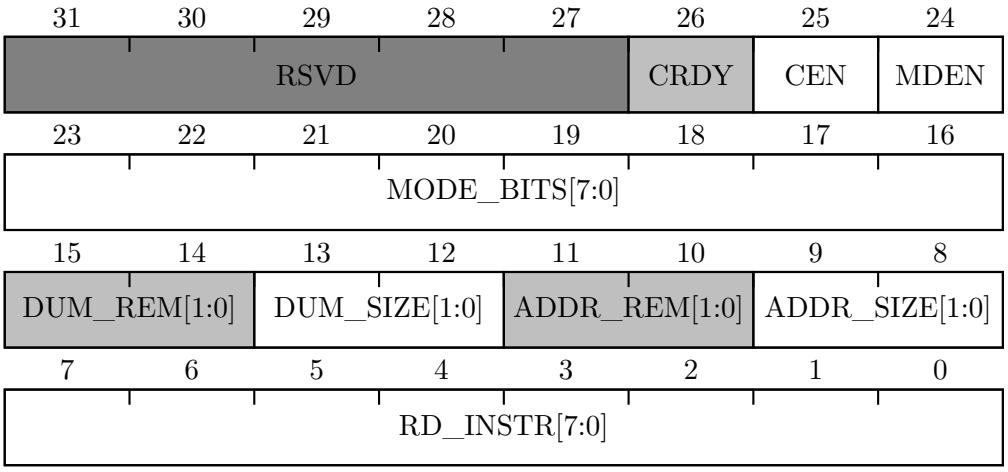
B.2.9 *Register 8 - 0x20 - Slave Select*



Bitfield	Access	Default value	Description
SPI_CS[31:0]	R/W	'hFFFFFFFF	These bits directly control the CS lines that are exposed out of the IP core. Up to 32 lines are supported, with each bit of this field corresponding to a single CS line. **This field can only be changed while MMIO mode is disabled (or unsupported) (MMEN = 'b0).**

B.2.10 *Register 9 - 0x24 - MMIO Configuration and Status 1*

This register is only available when the core is configured to support MMIO mode. Otherwise, accesses to this address lead to undefined behaviour. **This register can only be changed while MMIO mode is disabled (MMEN = 'b0).**

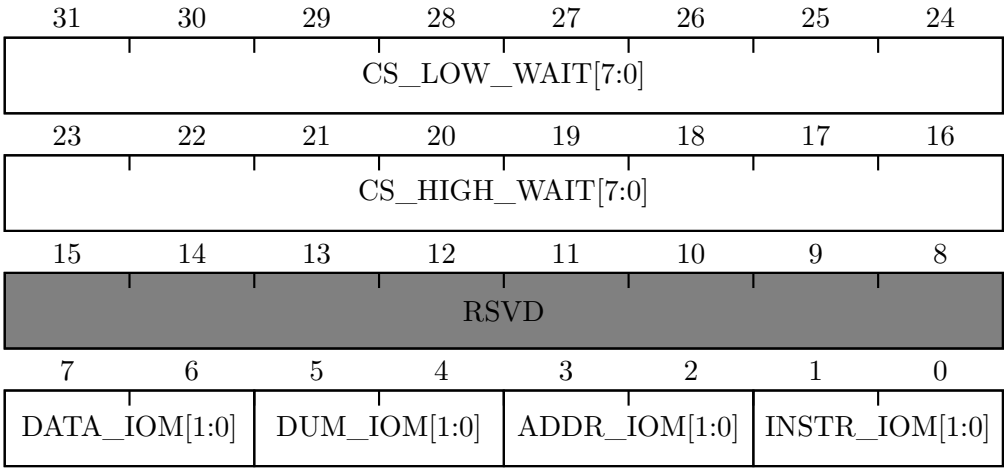


Bitfield	Access	Default value	Description
CRDY	R	'b0	This bit is set to 'b1 by the MMIO FSM once the first access has been made to the flash and it is placed into continuous read mode. Used for debug purposes.
CEN	R/W	'b0	Enables continuous read mode on the first access to the flash. This usually requires the correct mode bits to be configured (via MODE_BITS[7:0]) and enabled (via MDEN).

Bitfield	Access	Default value	Description
MODE_BITS[7:0]	R/W	'h00	Data to be sent to the flash during the mode bits phase of the transaction.
DUM_REM[1:0]	R	'b00	Remaining dummy bytes on the current transaction. Used for debug purposes.
DUM_SIZE[1:0]	R/W	'b00	Number of dummy bytes to be sent to the flash during each transaction: **'b00**: No dummy bytes. The dummy phase is skipped altogether. **'b01**: One dummy byte. This may correspond to 8, 4 or 2 SCK cycles, depending on the configured IO mode. **'b10**: Two dummy bytes. This may correspond to 16, 8 or 4 SCK cycles, depending on the configured IO mode. **'b11**: Three dummy bytes. This may correspond to 24, 12 or 6 SCK cycles, depending on the configured IO mode.
ADDR_REM[1:0]	R	'b00	Remaining address bytes on the current transaction. Used for debug purposes.
ADDR_SIZE[1:0]	R/W	'b00	Number of address bytes to be sent to the flash during each transaction: **'b00**: One address byte. **'b01**: Two address bytes. **'b10**: Three address bytes. **'b11**: Four address bytes.
RD_INSTR[7:0]	R/W	'h00	Data to be sent to the flash during the instruction phase of the transaction. This should be set according to the selected IO mode.

B.2.11 *Register 10 - 0x28 - MMIO Configuration and Status 2*

This register is only available when the core is configured to support MMIO mode. Otherwise, accesses to this address lead to undefined behaviour. **This register can only be changed while MMIO mode is disabled (MMEN = 'b0).**

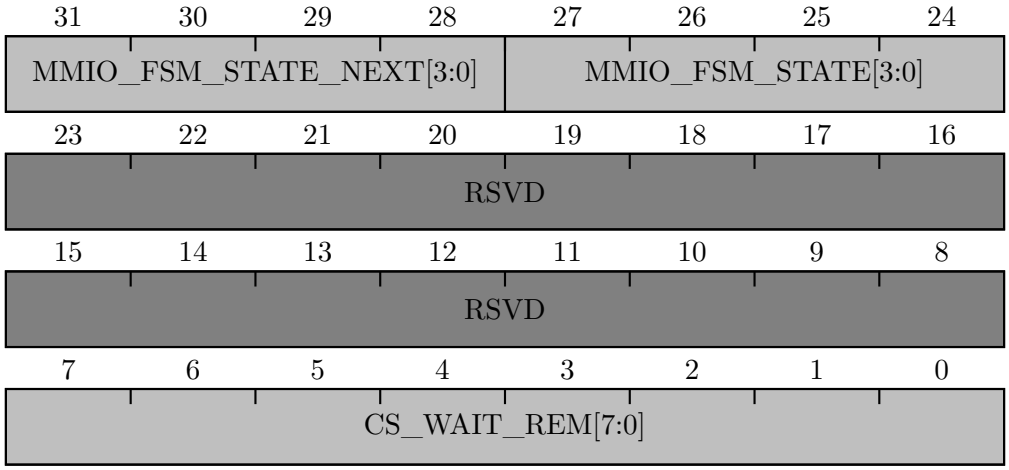


Bitfield	Access	Default value	Description
CS_LOW_WAIT[7:0]	R/W	'h00	Number of f_{aclk} cycles to wait after bringing CS low.
CS_HIGH_WAIT[7:0]	R/W	'h00	Number of f_{aclk} cycles to wait after bringing CS high.
DATA_IOM[1:0]	R/W	'b00	SPI IO mode to use during the data phase of transactions. The encoding is the same as IOMODE[1:0] (appendix B.2.2).
DUM_IOM[1:0]	R/W	'b00	SPI IO mode to use during the dummy phase of transactions. The encoding is the same as IOMODE[1:0] (appendix B.2.2).
ADDR_IOM[1:0]	R/W	'b00	SPI IO mode to use during the address phase of transactions. The encoding is the same as IOMODE[1:0] (appendix B.2.2).

Bitfield	Access	Default value	Description
INSTR_IOM[1:0]	R/W	'b00	SPI IO mode to use during the instruction phase of transactions. The encoding is the same as IOMODE[1:0] (appendix B.2.2).

B.2.12 *Register 11 - 0x2C - MMIO Configuration and Status 3*

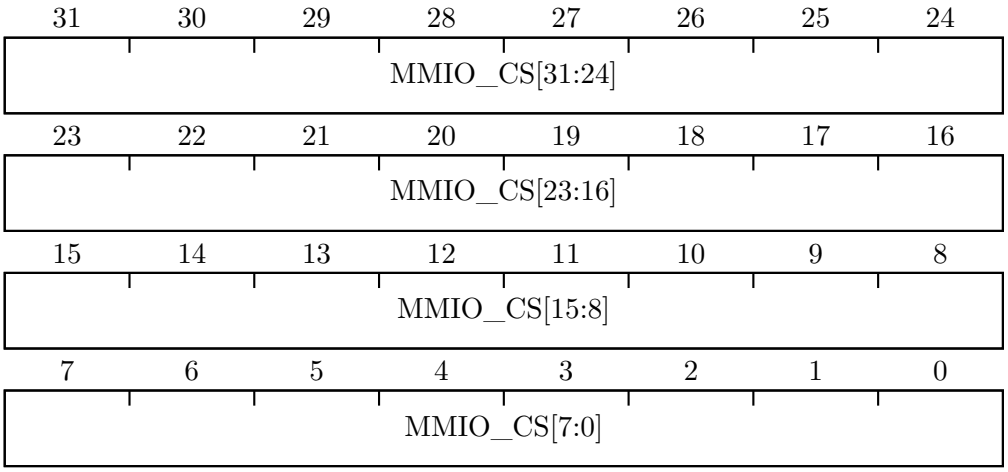
This register is only available when the core is configured to support MMIO mode. Otherwise, accesses to this address lead to undefined behaviour. **This register can only be changed while MMIO mode is disabled (MMEN = 'b0).**



Bitfield	Access	Default value	Description
MMIO_FSM_STATE_NEXT[3:0]	R	'b0000	Next SPI MMIO FSM state. Used for debug purposes.
MMIO_FSM_STATE[3:0]	R	'b0000	Current SPI MMIO FSM state. Used for debug purposes.
CS_WAIT_REM[7:0]	R	'h00	Remaining cycles of the CS wait counter. Used for debug purposes.

B.2.13 *Register 12 - 0x30 - MMIO Slave Select Mask*

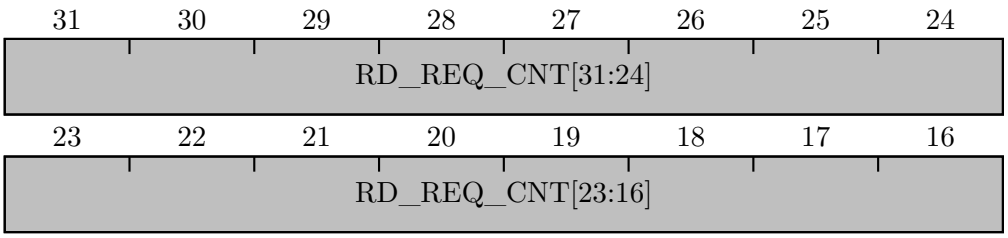
This register is only available when the core is configured to support MMIO mode. Otherwise, accesses to this address lead to undefined behaviour. **This register can only be changed while MMIO mode is disabled (MMEN = 'b0).**

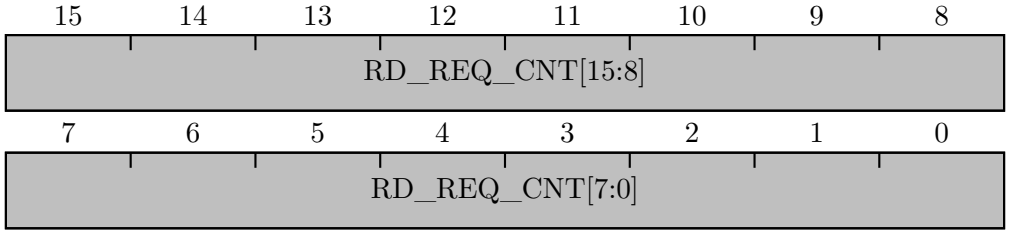


Bitfield	Access	Default value	Description
MMIO_CS[31:0]	R/W	'h00000000	These bits define the mask of bits that will be toggled when automatically asserting or de-asserting the SPI CS lines.

B.2.14 *Register 13 - 0x34 - MMIO Diagnostics 1*

This register is only available when the core is configured to support MMIO mode. Otherwise, accesses to this address lead to undefined behaviour. **This register can only be changed while MMIO mode is disabled (MMEN = 'b0).**

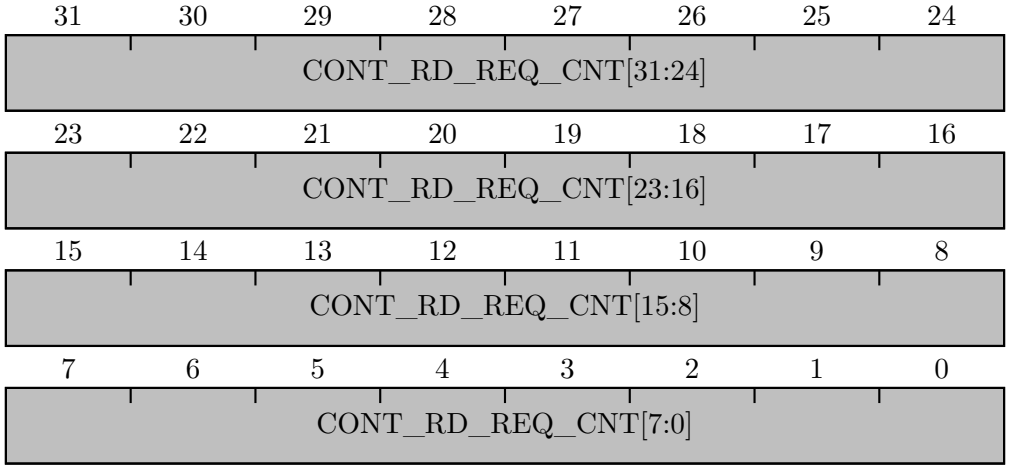




Bitfield	Access	Default value	Description
RD_REQ_CNT[31:0]	R/W	'h00000000	This bitfield represents the number of read transactions received on the AXI4-Full interface which directly maps to the SPI flash address space. The counter resets when this register is read. Reading this register also buffers the current value of CONT_RD_REQ_CNT[31:0] and clear it afterwards.

B.2.15 *Register 14 - 0x38 - MMIO Diagnostics 2*

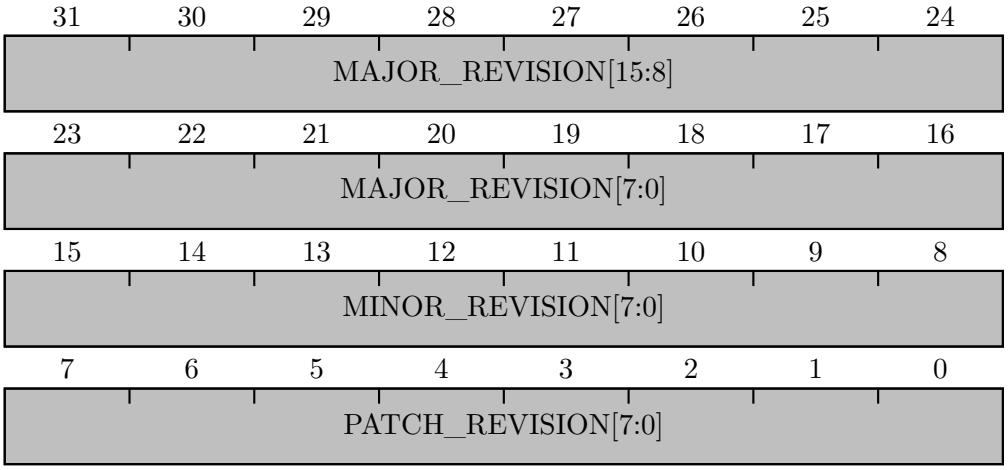
This register is only available when the core is configured to support MMIO mode. Otherwise, accesses to this address lead to undefined behaviour. **This register can only be changed while MMIO mode is disabled (MMEN = 'b0).**



Bitfield	Access	Default value	Description
CONT_RD_REQ_CNT[31:0]	R/W	'h00000000	This bitfield represents the number of read transactions received on the AXI4-Full interface that were answered without raising the CS line, i.e., the address was contiguous. This register reflects the buffered value which is saved when RD_REQ_CNT[31:0] is read, so it must be read first.

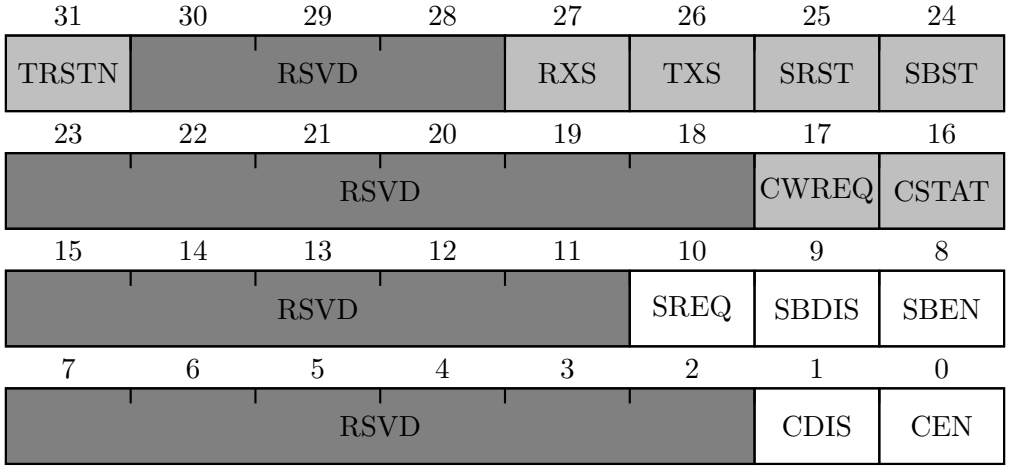
B.3 AXI RF TIMESTAMPING IP REGISTER DOCUMENTATION

B.3.1 *Register 0 - 0x00 - IP Version*



Bitfield	Access	Default value	Description
MAJOR_REVISION[15:0]	R	'h0001	IP module major revision v[**MAJ**].[MIN].[PATCH]
MINOR_REVISION[7:0]	R	'h00	IP module minor revision v[MAJ].[**MIN**].[PATCH]
PATCH_REVISION[7:0]	R	'h00	IP module patch revision v[MAJ].[MIN].[**PATCH**]

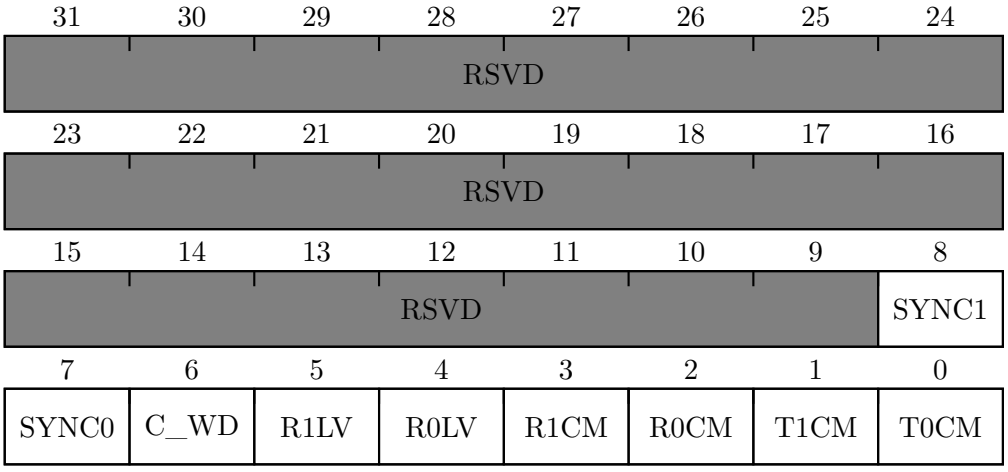
B.3.2 *Register 1 - 0x04 - Status and Control*



Bitfield	Access	Default value	Description
TRSTN	R	'b0	RF sample clock domain synchronous reset (inverted), synchronized to the AXI-Lite clock domain. Used for debug purposes.
RXS	R	'b0	This bit indicates whether the internal clock gating logic is synchronized to the RX data valid edge.
TXS	R	'b0	This bit indicates whether the internal clock gating logic is synchronized to the TX data valid edge.
SRST	R	'b0	This bit is high when a pending clock synchronization request is pending.
SBST	R	'b0	This bit indicates whether the clock synchronization mechanism is currently bypassed ('b1) or not ('b0).

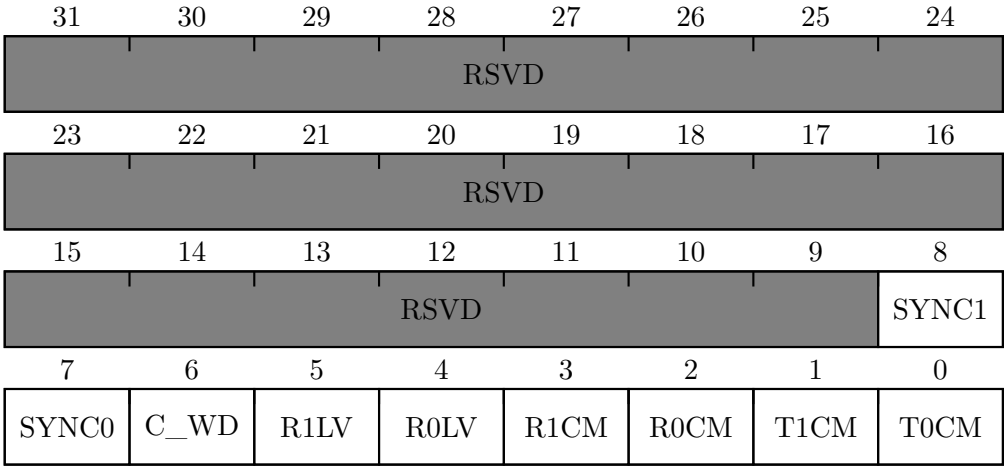
Bitfield	Access	Default value	Description
CWREQ	R	'b0	This bit is high when a pending counter write request is pending. Software should not write to CTR[63:0] while this bit is 'b1.
CSTAT	R	'b0	This bit indicates whether the free-running counter is enabled ('b1) or disabled ('b0).
SREQ	R/W	'b0	Write 'b1 to place a request for clock synchronization. This bit always reads as 'b0.
SBDIS	R/W	'b0	Write 'b1 to disable the clock synchronization bypass. This bit always reads as 'b0.
SBEN	R/W	'b0	Write 'b1 to enable the clock synchronization bypass. This bit always reads as 'b0.
CDIS	R/W	'b0	Write 'b1 to disable the free-running counter. This bit always reads as 'b0.
CEN	R/W	'b0	Write 'b1 to enable the free-running counter. This bit always reads as 'b0.

B.3.3 *Register 2 - 0x08 - IRQ Enable*



Bitfield	Access	Default value	Description
SYNC1	R/W	'b0	IRQ mask for CDC synchronization handshake completion from *ts_clk* to *aclk* ('b0 = Masked, 'b1 = Unmasked). Used for debug purposes.
SYNC0	R/W	'b0	IRQ mask for CDC synchronization handshake completion from *aclk* to *ts_clk* ('b0 = Masked, 'b1 = Unmasked). Used for debug purposes.
C_WD	R/W	'b0	IRQ mask for counter write completion ('b0 = Masked, 'b1 = Unmasked).
R1LV	R/W	'b0	IRQ mask for when valid data is latched into the RX1 latch ('b0 = Masked, 'b1 = Unmasked).
R0LV	R/W	'b0	IRQ mask for when valid data is latched into the RX0 latch ('b0 = Masked, 'b1 = Unmasked).
R1CM	R/W	'b0	IRQ mask for when a match occurs between the counter value and the RX1 timestamp ('b0 = Masked, 'b1 = Unmasked).
R0CM	R/W	'b0	IRQ mask for when a match occurs between the counter value and the RX0 timestamp ('b0 = Masked, 'b1 = Unmasked).
T1CM	R/W	'b0	IRQ mask for when a match occurs between the counter value and the TX1 timestamp ('b0 = Masked, 'b1 = Unmasked).
T0CM	R/W	'b0	IRQ mask for when a match occurs between the counter value and the TX0 timestamp ('b0 = Masked, 'b1 = Unmasked).

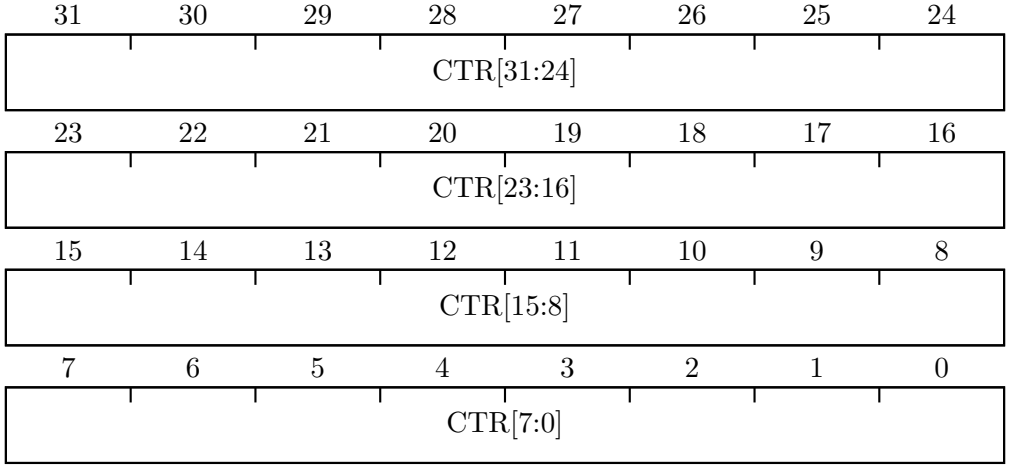
B.3.4 *Register 3 - 0x0C - IRQ Pending*



Bitfield	Access	Default value	Description
SYNC1	R/W1TC	'b0	CDC synchronization handshake from *ts_clk* to *aclk* is complete. Write 1 to clear the pending interrupt flag. Used for debug purposes.
SYNC0	R/W1TC	'b0	CDC synchronization handshake from *aclk* to *ts_clk* is complete. Write 1 to clear the pending interrupt flag. Used for debug purposes.
C_WD	R/W1TC	'b0	Counter write completion. Write 1 to clear the pending interrupt flag.
R1LV	R/W1TC	'b0	Valid data is latched into the RX1 latch. Write 1 to clear the pending interrupt flag.
R0LV	R/W1TC	'b0	Valid data is latched into the RX0 latch. Write 1 to clear the pending interrupt flag.
R1CM	R/W1TC	'b0	Match between the counter value and the RX1 timestamp occurred. Write 1 to clear the pending interrupt flag.
R0CM	R/W1TC	'b0	Match between the counter value and the RX0 timestamp occurred. Write 1 to clear the pending interrupt flag.
T1CM	R/W1TC	'b0	Match between the counter value and the TX1 timestamp occurred. Write 1 to clear the pending interrupt flag

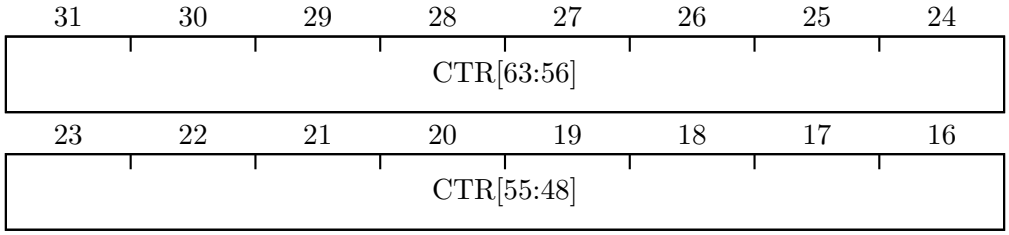
Bitfield	Access	Default value	Description
T0CM	R/W	'b0	Match between the counter value and the TX0 timestamp occurred. Write 1 to clear the pending interrupt flag

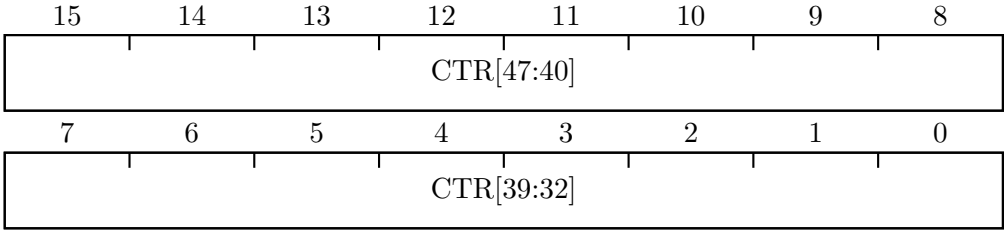
B.3.5 *Register 4 - 0x10 - Sample counter low*



Bitfield	Access	Default value	Description
CTR[31:0]	R/W	'h00000000	Least significant bytes of the free-running sample counter. Reading this register also buffers the current value of CTR[63:32]. Writing to this register buffers the written value to be used when CTR[63:32] is written.

B.3.6 *Register 5 - 0x14 - Sample counter high*





Bitfield	Access	Default value	Description
CTR[63:32]	R/W	'h00000000	Most significant bytes of the free-running sample counter. This register reflects the buffered value which is saved when CTR[31:0] is read, so it must be read first. Writing to this register triggers the actual counter value writing operation, together with the least significant bytes saved when CTR[31:0] was written, so it must be written first.

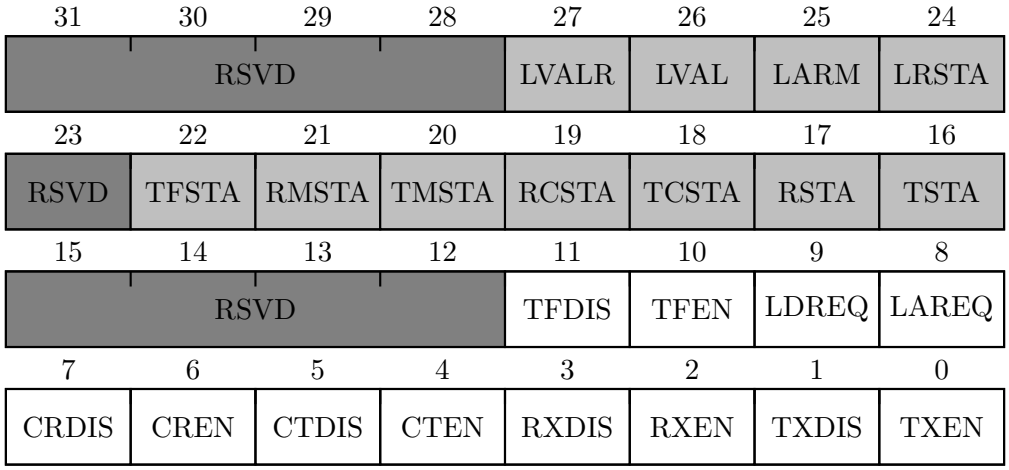
B.3.7 *Register 6 - 0x20 - Multi-channel Control and Status*

This register has the same layout as registers 8 and 16 (appendix B.3.9). Writing to this register has the same effect as writing to both of these registers atomically. Read-only bits read as 'b1 as long as at least one of the corresponding bits in registers 8 or 16 are set.

B.3.8 *Register 7 - 0x24 - Multi-channel Override Control*

This register has the same layout as registers 9 and 17 (appendix B.3.10). Writing to this register has the same effect as writing to both of these registers atomically. This register always reads as 'h00000000.

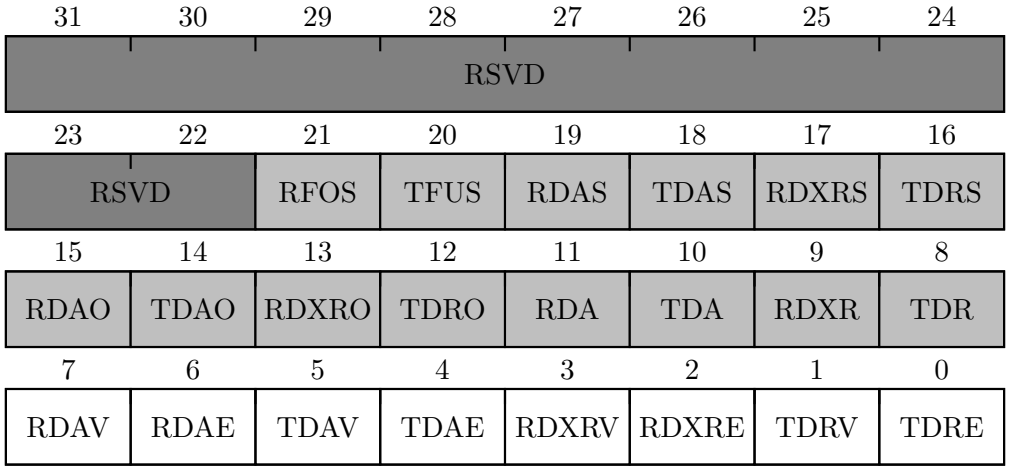
B.3.9 *Registers 8 and 16 - 0x40 + n * 0x20 - Channel n Control and Status*



Bitfield	Access	Default value	Description
LVALR	R	'b0	This bit indicates whether the RX latch invalidation process is ongoing ('b1) or not ('b0). Used for debug purposes.
LVAL	R	'b0	This bit indicates whether the RX latch has valid data ('b1) or not ('b0).
LARM	R	'b0	This bit indicates whether the RX latch is armed ('b1) or not ('b0).
LRSTA	R	'b0	This bit indicates whether a RX latch arm request is pending ('b1) or not ('b0).
TFSTA	R	'b0	This bit indicates the TX flush logic is enabled ('b1) or not ('b0).
RMSTA	R	'b0	This bit indicates the status of the manual RX control. This may not correspond to the actual RX status.
TMSTA	R	'b0	This bit indicates the status of the manual TX control. This may not correspond to the actual TX status.
RCSTA	R	'b0	This bit indicates the status of the RX counter.
TCSTA	R	'b0	This bit indicates the status of the TX counter.

Bitfield	Access	Default value	Description
RSTA	R	'b0	This bit indicates the current status of the RX control. This always traduces the actual RX status.
TSTA	R	'b0	This bit indicates the current status of the TX control. This always traduces the actual TX status.
TFDIS	R/W	'b0	Write 'b1 to disable the TX flush logic. This bit always reads as 'b0.
TFEN	R/W	'b0	Write 'b1 to enable the TX flush logic. This bit always reads as 'b0.
LDREQ	R/W	'b0	Write 'b1 to request the RX latch to disarm. This bit always reads as 'b0.
LAREQ	R/W	'b0	Write 'b1 to request the RX latch to arm. This bit always reads as 'b0.
CRDIS	R/W	'b0	Write 'b1 to disable the RX counter match. This bit always reads as 'b0.
CREN	R/W	'b0	Write 'b1 to enable the RX counter match. This bit always reads as 'b0.
CTDIS	R/W	'b0	Write 'b1 to disable the TX counter match. This bit always reads as 'b0.
CTEN	R/W	'b0	Write 'b1 to enable the TX counter match. This bit always reads as 'b0.
RXDIS	R/W	'b0	Write 'b1 to manually disable the RX. This bit always reads as 'b0. **RX manual control is only available while the RX counter is disabled (RCSTA = 'b0).**
RXEN	R/W	'b0	Write 'b1 to manually enable the RX. This bit always reads as 'b0. **RX manual control is only available while the RX counter is disabled (RCSTA = 'b0).**
TXDIS	R/W	'b0	Write 'b1 to manually disable the TX. This bit always reads as 'b0. **TX manual control is only available while the TX counter is disabled (TCSTA = 'b0).**
TXEN	R/W	'b0	Write 'b1 to manually enable the TX. This bit always reads as 'b0. **TX manual control is only available while the TX counter is disabled (TCSTA = 'b0).**

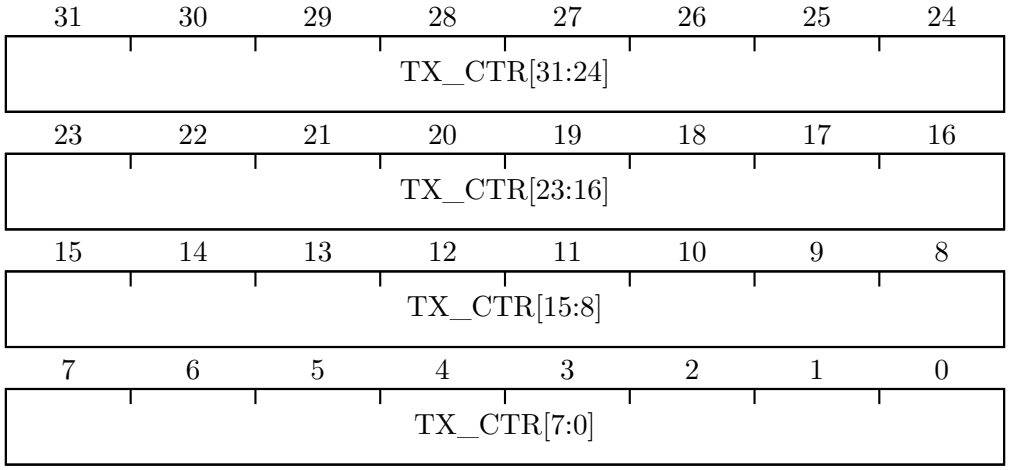
B.3.10 *Registers 9 and 17 - 0x44 + n * 0x20 - Channel n Override Control*



Bitfield	Access	Default value	Description
RFOS	R	'b0	This bit indicates whether an overflow occurred in the RX FIFO since the last time this register was read. This bit clears when read. Used for debug purposes.
TFUS	R	'b0	This bit indicates whether an underflow occurred in the TX FIFO since the last time this register was read. This bit clears when read. Used for debug purposes.
RDAS	R	'b0	This bit indicates whether the *tready* signal of the RX AXI-Stream has been high since the last time this register was read. This bit clears when read.
TDAS	R	'b0	This bit indicates whether the *tready* signal of the TX AXI-Stream has been high since the last time this register was read. This bit clears when read.
RDXRS	R	'b0	This bit indicates whether there has been a RX DMA transfer request since the last time this register was read. This bit clears when read.
TDRS	R	'b0	This bit indicates whether there has been valid data on the TX DMA since the last time this register was read. This bit clears when read.

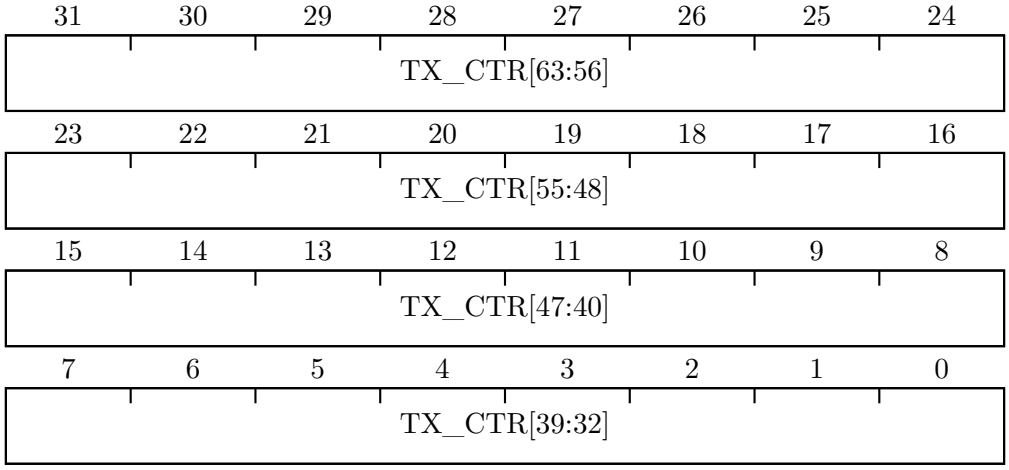
Bitfield	Access	Default value	Description
RDAO	R	'b0	This bit reflects the overridden RX AXI-Stream *tready* signal value.
TDAO	R	'b0	This bit reflects the overridden TX AXI-Stream *tready* signal value.
RDXRO	R	'b0	This bit reflects the overridden RX DMA transfer request signal value.
TDRO	R	'b0	This bit reflects the overridden TX DMA data ready signal value.
RDA	R	'b0	This bit reflects the non-overridden RX AXI-Stream *tready* signal value.
TDA	R	'b0	This bit reflects the non-overridden TX AXI-Stream *tready* signal value.
RDXR	R	'b0	This bit reflects the non-overridden RX DMA transfer request signal value.
TDR	R	'b0	This bit reflects the non-overridden TX DMA data valid signal value.
RDAV	R/W	'b0	This bit sets the value to override the RX AXI-Stream *tready* signal with.
RDAE	R/W	'b0	This bit enabled ('b1) or disables ('b0) overriding the RX AXI-Stream *tready* signal value.
TDAV	R/W	'b0	This bit sets the value to override the TX AXI-Stream *tready* signal with.
TDAE	R/W	'b0	This bit enabled ('b1) or disables ('b0) overriding the TX AXI-Stream *tready* signal value.
RDXRV	R/W	'b0	This bit sets the value to override the RX DMA transfer request signal with.
RDXRE	R/W	'b0	This bit enabled ('b1) or disables ('b0) overriding the RX DMA transfer request signal value.
TDRV	R/W	'b0	This bit sets the value to override the TX DMA data valid signal with.
TDRE	R/W	'b0	This bit enabled ('b1) or disables ('b0) overriding the RX DMA data valid signal value.

B.3.11 *Register 10 and 18 - 0x48 + n * 0x20 - Channel n TX counter low*



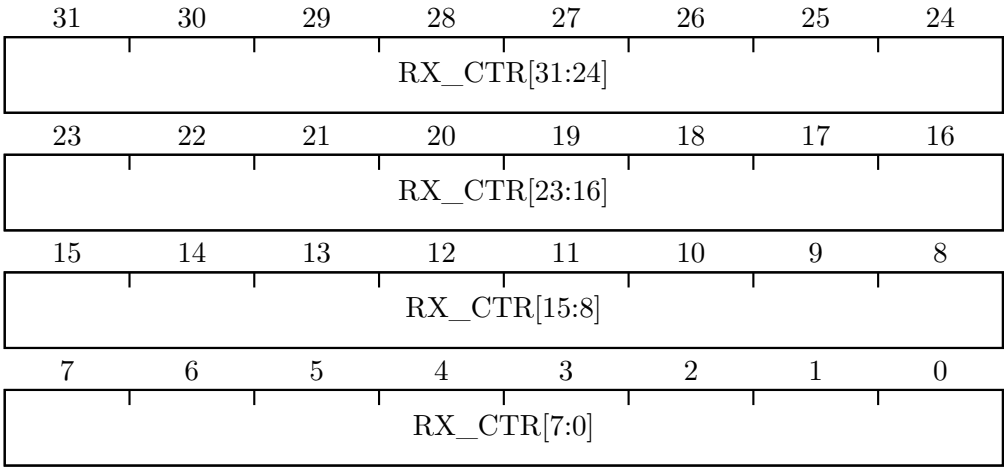
Bitfield	Access	Default value	Description
TX_CTR[31:0]	R/W	'h00000000	Least significant bytes of the TX counter match value which triggers TX state change. Reading this register also buffers the current value of TX_CTR[63:32]. Writing to this register buffers the written value to be used when TX_CTR[63:32] is written. **This field may only be written while the TX counter is disabled (TCSTA = 'b0).**

B.3.12 *Register 11 and 19 - 0x4C + n * 0x20 - Channel n TX counter high*



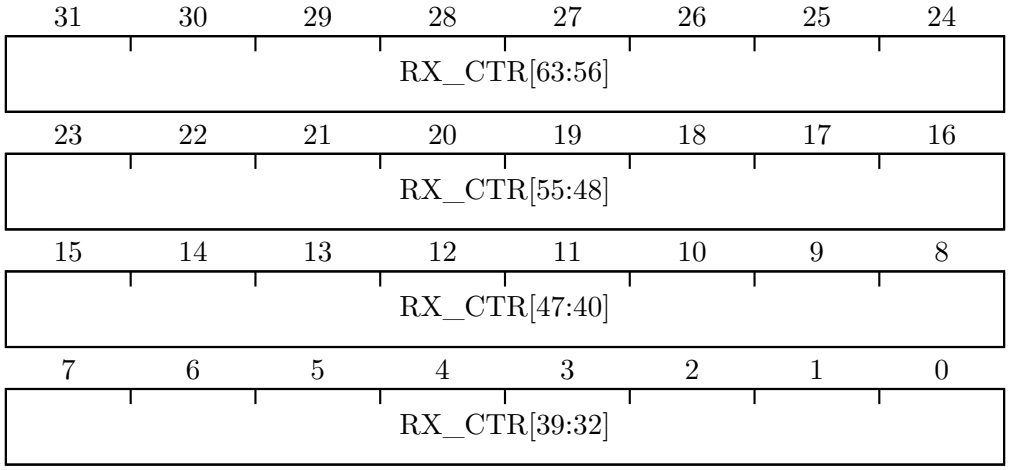
Bitfield	Access	Default value	Description
TX_CTR[63:32]	R/W	'h00000000	Most significant bytes of the TX counter match value which triggers TX state change. This register reflects the buffered value which is saved when TX_CTR[31:0] is read, so it must be read first. Writing to this register triggers the actual counter value writing operation, together with the least significant bytes saved when TX_CTR[31:0] was written, so it must be written first. **This field may only be written while the TX counter is disabled (TCSTA = 'b0).**

B.3.13 *Register 12 and 20 - 0x50 + n * 0x20 - Channel n RX counter low*



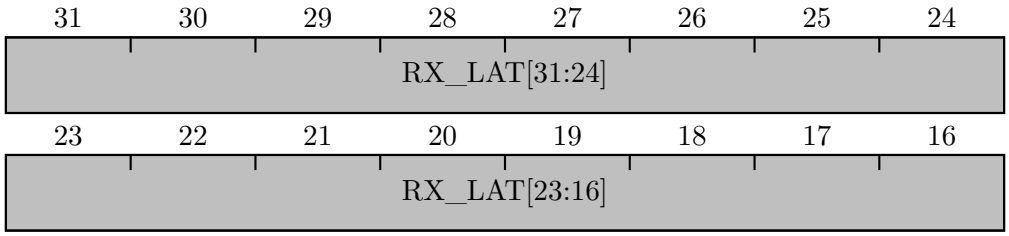
Bitfield	Access	Default value	Description
RX_CTR[31:0]	R/W	'h00000000	Least significant bytes of the RX counter match value which triggers RX state change. Reading this register also buffers the current value of RX_CTR[63:32]. Writing to this register buffers the written value to be used when RX_CTR[63:32] is written. **This field may only be written while the RX counter is disabled (RCSTA = 'b0).**

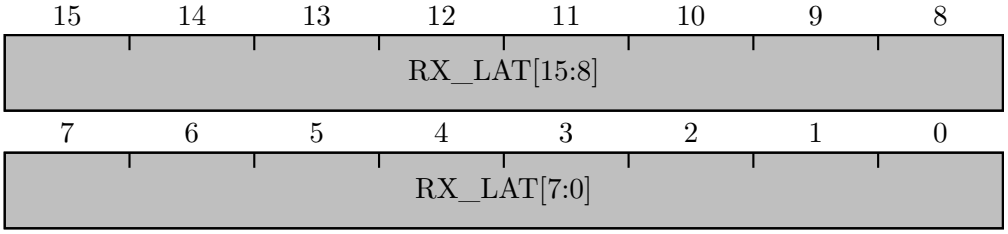
B.3.14 *Register 13 and 21 - 0x54 + n * 0x20 - Channel n RX counter high*



Bitfield	Access	Default value	Description
RX_CTR[63:32]	R/W	'h00000000	Most significant bytes of the RX counter match value which triggers RX state change. This register reflects the buffered value which is saved when RX_CTR[31:0] is read, so it must be read first. Writing to this register triggers the actual counter value writing operation, together with the least significant bytes saved when RX_CTR[31:0] was written, so it must be written first. **This field may only be written while the RX counter is disabled (RCSTA = 'b0).**

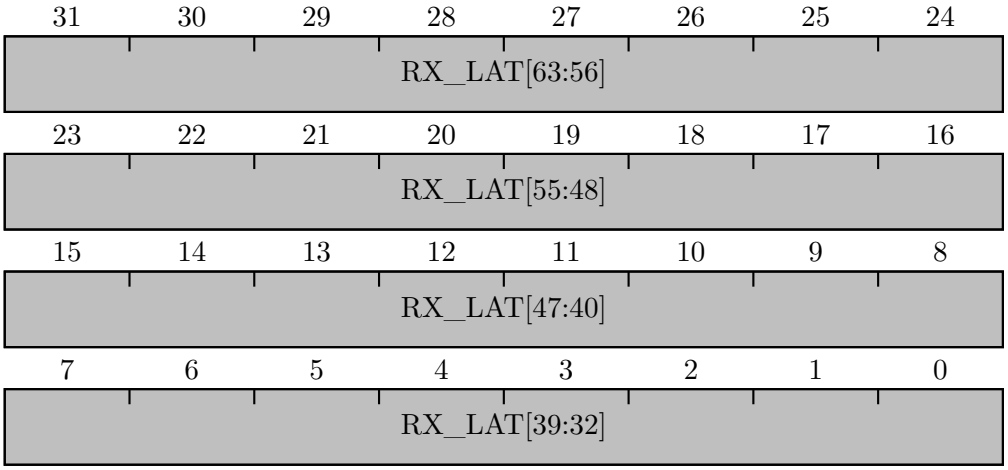
B.3.15 *Register 14 and 22 - 0x58 + n * 0x20 - Channel n RX counter latch low*





Bitfield	Access	Default value	Description
RX_LAT[31:0]	R	'h00000000	Least significant bytes of the latched RX counter match value. Reading this register also buffers the current value of RX_LAT[63:32] and clears the latch valid flag (LVAL). **The data on this register is undefined unless LVAL = 'b1 in the Control and Status Register (appendix B.3.9).**

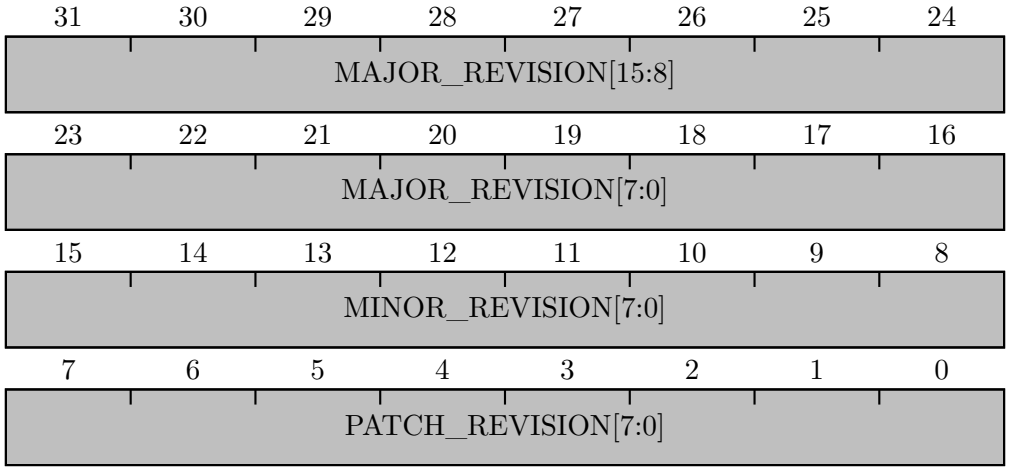
B.3.16 *Register 15 and 23 - 0x5C + n * 0x20 - Channel n RX counter latch high*



Bitfield	Access	Default value	Description
RX_LAT[63:32]	R	'h00000000	Most significant bytes of the latched RX counter match value. This register reflects the buffered value which is saved when RX_LAT[31:0] is read, so it must be read first.

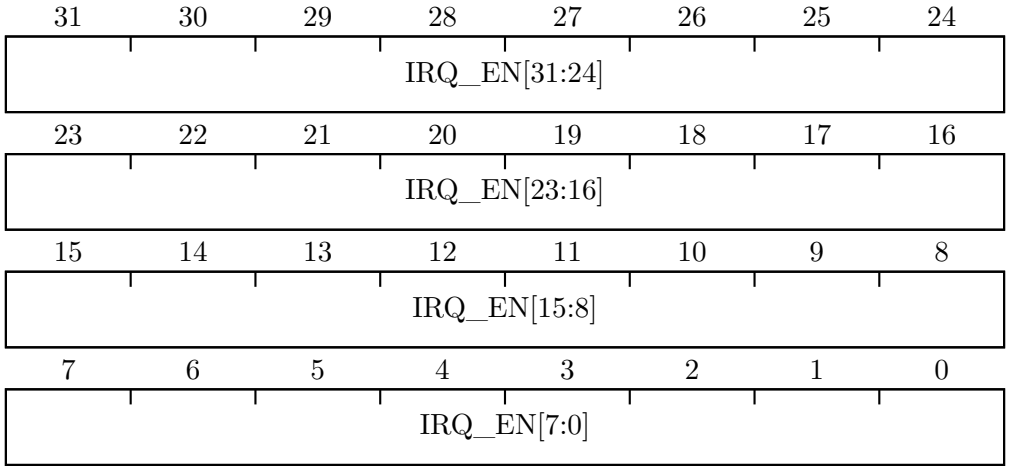
B.4 AXI IRQ CONTROLLER IP REGISTER DOCUMENTATION

B.4.1 *Register 0 - 0x00 - IP Version*



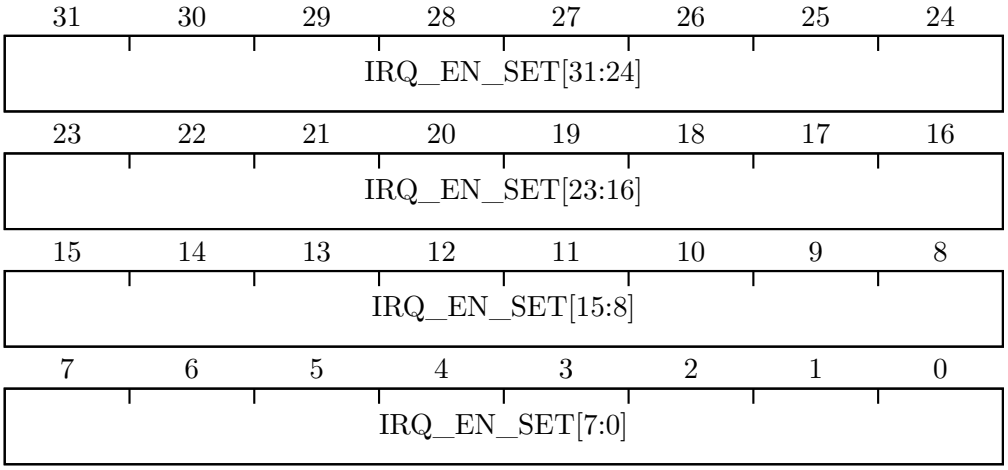
Bitfield	Access	Default value	Description
MAJOR_REVISION[15:0]	R	'h0001	IP module major revision v[**MAJ**].[MIN].[PATCH]
MINOR_REVISION[7:0]	R	'h00	IP module minor revision v[MAJ].[**MIN**].[PATCH]
PATCH_REVISION[7:0]	R	'h00	IP module patch revision v[MAJ].[MIN].[**PATCH**]

B.4.2 *Register 1 - 0x04 - IRQ Enable*



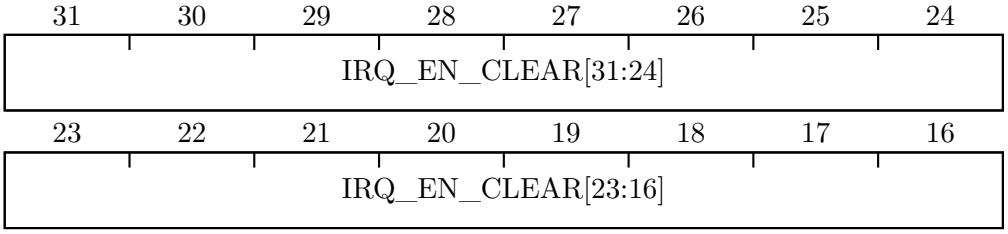
Bitfield	Access	Default value	Description
IRQ_EN[31:0]	R/W	'h00000000	These bits mask the generation of interrupts. Each bit controls a single IRQ line. A set bit ('b1) enables the corresponding line, while a clear bit ('b0) disables it.

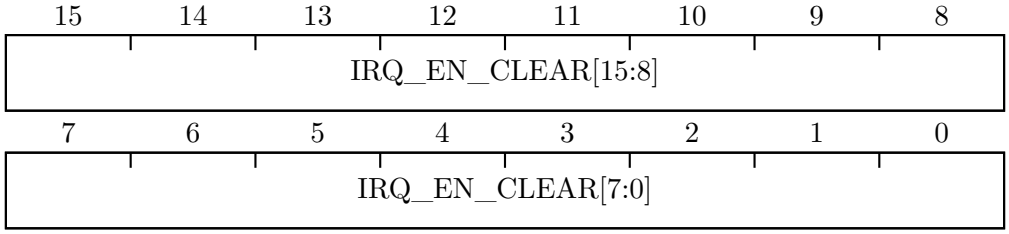
B.4.3 *Register 2 - 0x08 - IRQ Enable (W1TS)*



Bitfield	Access	Default value	Description
IRQ_EN_SET[31:0]	R/W1	'h00000000	Writing one to any of these bits sets the corresponding IRQ_EN bit. Writing zero has no effect. The read data is equal to the IRQ enable register (appendix B.4.2).

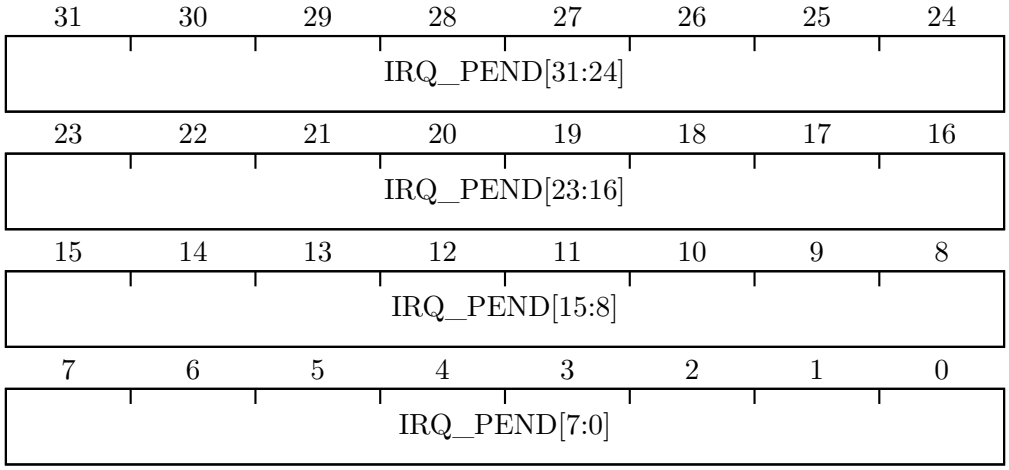
B.4.4 *Register 3 - 0x0C - IRQ Enable (W1TC)*





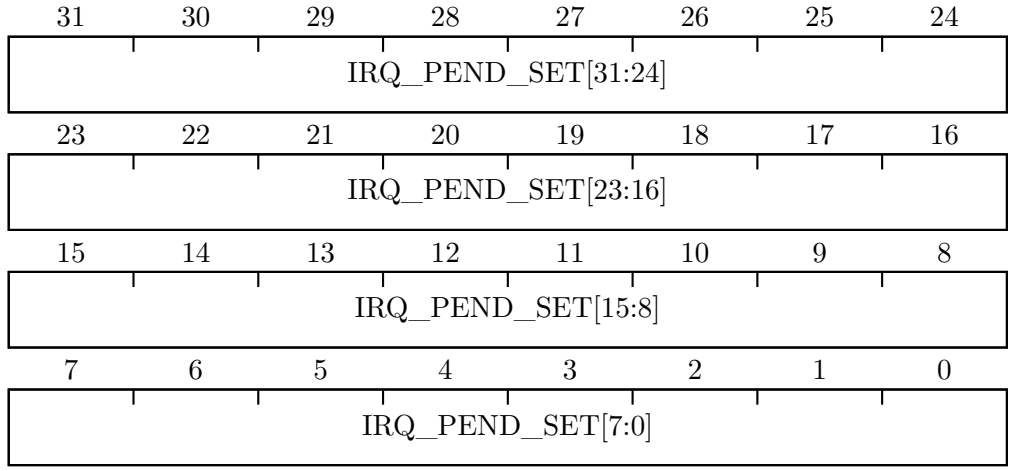
Bitfield	Access	Default value	Description
IRQ_EN_CLEAR[31:0]	R/W1	'h00000000	Writing one to any of these bits clears the corresponding IRQ_EN bit. Writing zero has no effect. The read data is equal to the IRQ enable register (appendix B.4.2).

B.4.5 *Register 4 - 0x10 - IRQ Pensing*



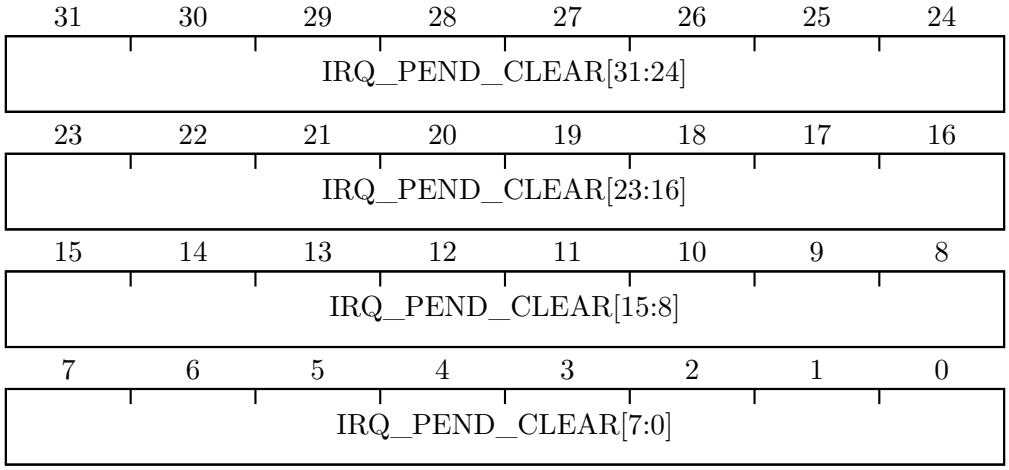
Bitfield	Access	Default value	Description
IRQ_PEND[31:0]	R/W	'h00000000	These bits indicate if any interrupt is pending. Each bit is associated with a single IRQ line. A set bit ('b1) enables the corresponding interrupt is pending, while a clear bit ('b0) indicates it is not pending. These bits are set when the configured interrupt is triggered, but must be manually cleared. Atomic access registers are useful for this (see appendices B.4.6 and B.4.7)

B.4.6 *Register 5 - 0x14 - IRQ Pending (W1TS)*



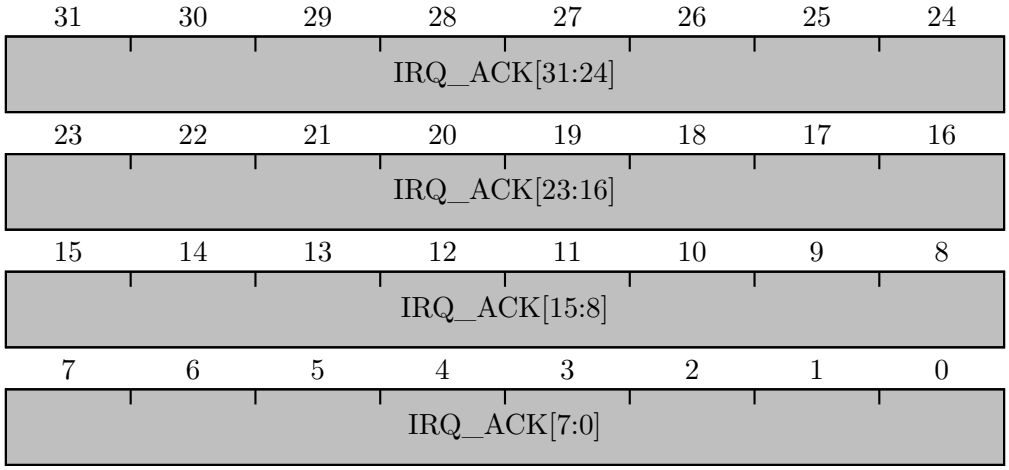
Bitfield	Access	Default value	Description
IRQ_PEND_SET[31:0]	R/W1	'h00000000	Writing one to any of these bits sets the corresponding IRQ_PEND bit. Writing zero has no effect. The read data is equal to the IRQ pending register (appendix B.4.5).

B.4.7 *Register 6 - 0x18 - IRQ Pending (W1TC)*



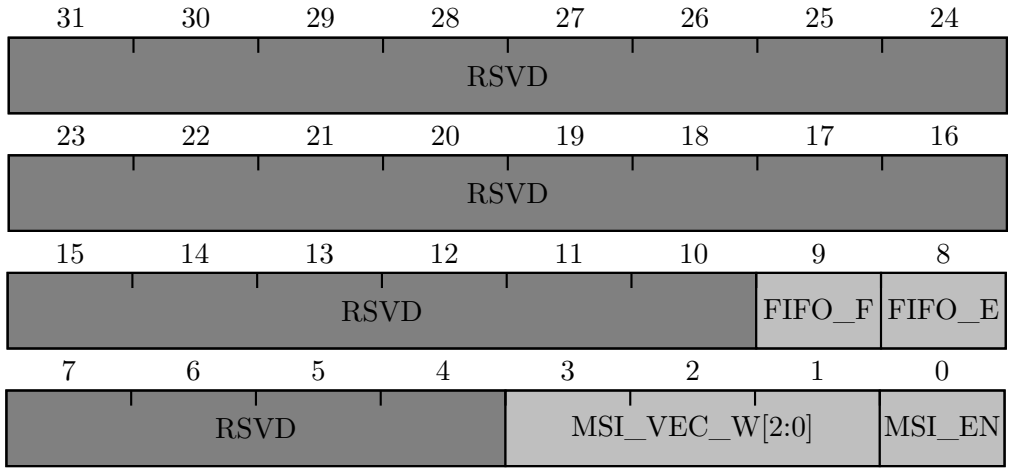
Bitfield	Access	Default value	Description
IRQ_PEND_CLEAR[31:0]	R/W1	'h00000000	Writing one to any of these bits clears the corresponding IRQ_PEND bit. Writing zero has no effect. The read data is equal to the IRQ pending register (appendix B.4.5).

B.4.8 *Register 7 - 0x1C - IRQ Acknowledged*



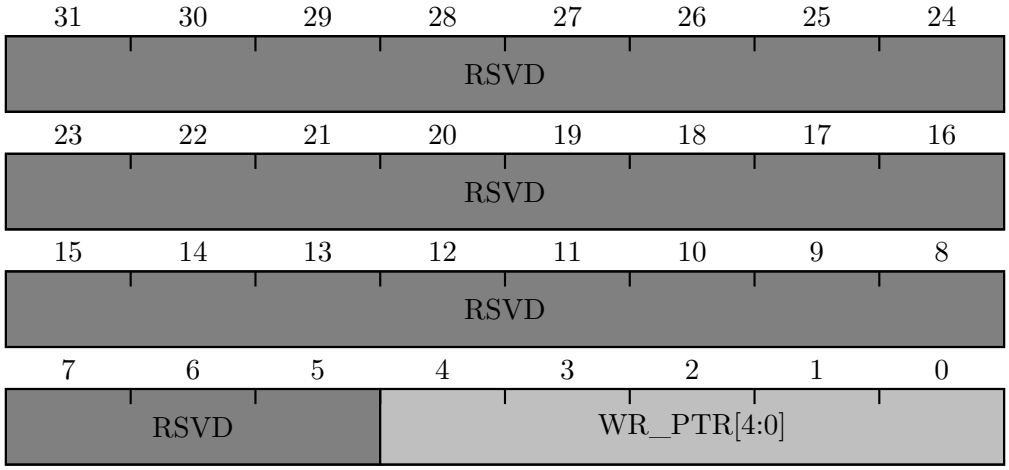
Bitfield	Access	Default value	Description
IRQ_ACK[31:0]	R	'h00000000	These bits indicate if an interrupt has been acknowledged by the core and sent out to the configured destination. Used for debug purposes.

B.4.9 *Register 8 - 0x40 - PCIe MSI Status*



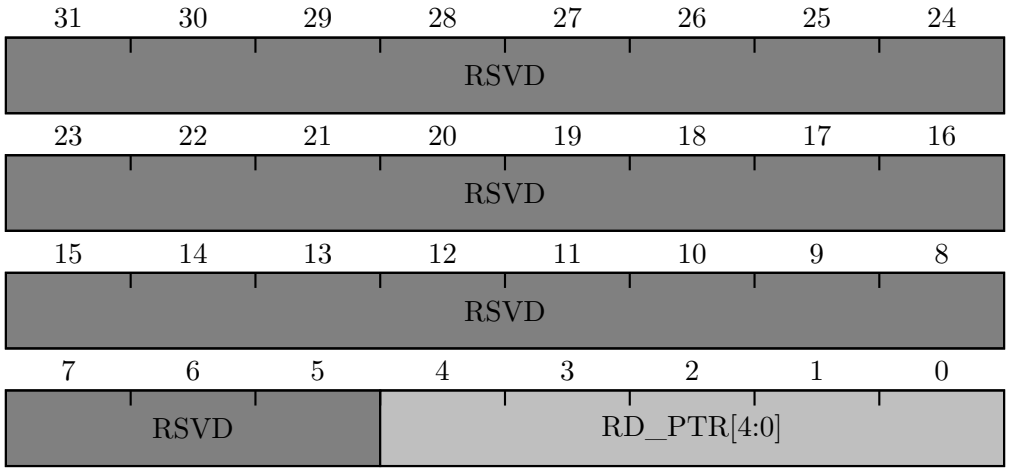
Bitfield	Access	Default value	Description
FIFO_F	R	'b0	Whether the MSI request FIFO is full or not.
FIFO_E	R	'b0	Whether the MSI request FIFO is empty or not.
MSI_VEC_W[2:0]	R	'b000	The number of MSI vectors available, as reported by the PCIe core. **'b000**: 1 MSI vector available. **'b001**: 2 MSI vectors available. **'b010**: 4 MSI vectors available. **'b011**: 8 MSI vectors available. **'b100**: 16 MSI vectors available. **'b101**: 32 MSI vectors available. **'b110**: Reserved. **'b111**: Reserved.
MSI_EN	R	'b0	Whether the PCIe core reported that MSIs are supported.

B.4.10 *Register 9 - 0x44 - PCIe MSI request FIFO write pointer*



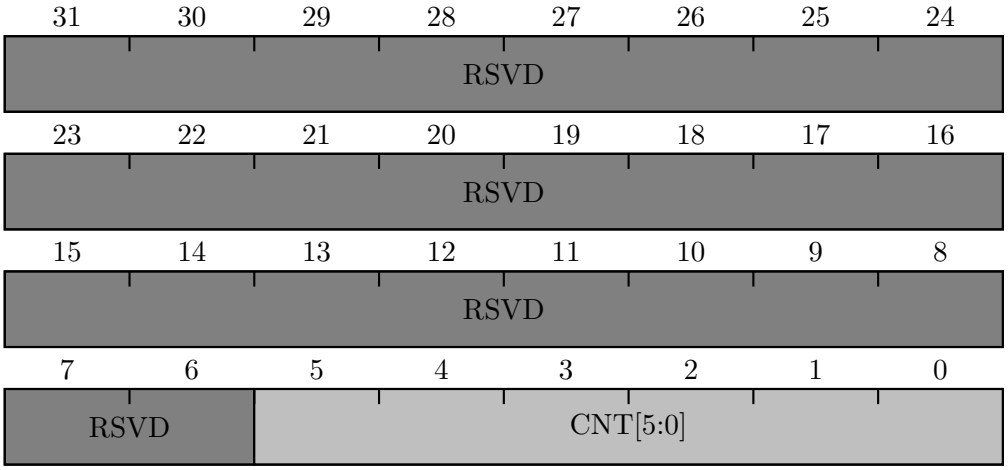
Bitfield	Access	Default value	Description
WR_PTR[4:0]	R	'b00000	The current FIFO write pointer. Used for debug purposes.

B.4.11 *Register 10 - 0x48 - PCIe MSI request FIFO read pointer*



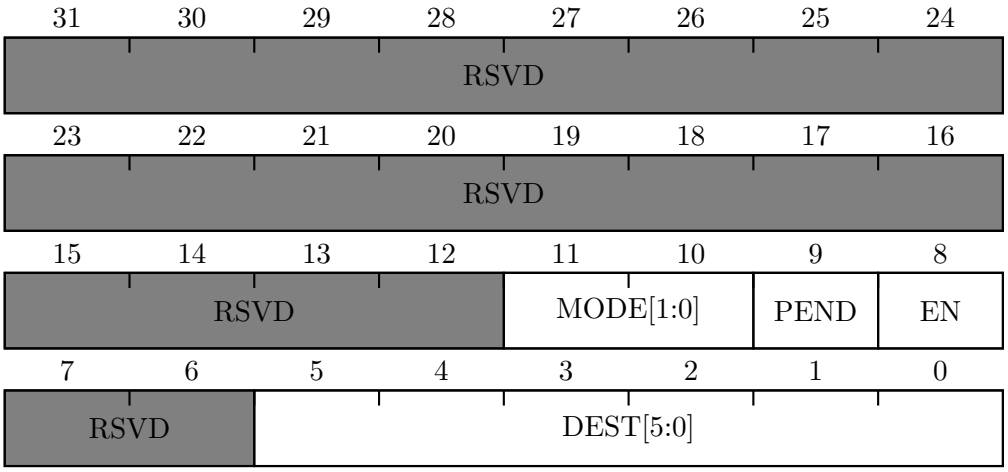
Bitfield	Access	Default value	Description
RD_PTR[4:0]	R	'b00000	The current FIFO read pointer. Used for debug purposes.

B.4.12 *Register 10 - 0x4C - PCIe MSI request FIFO count*



Bitfield	Access	Default value	Description
CNT[5:0]	R	'b000000	The number of valid entries currently in the FIFO. Used for debug purposes.

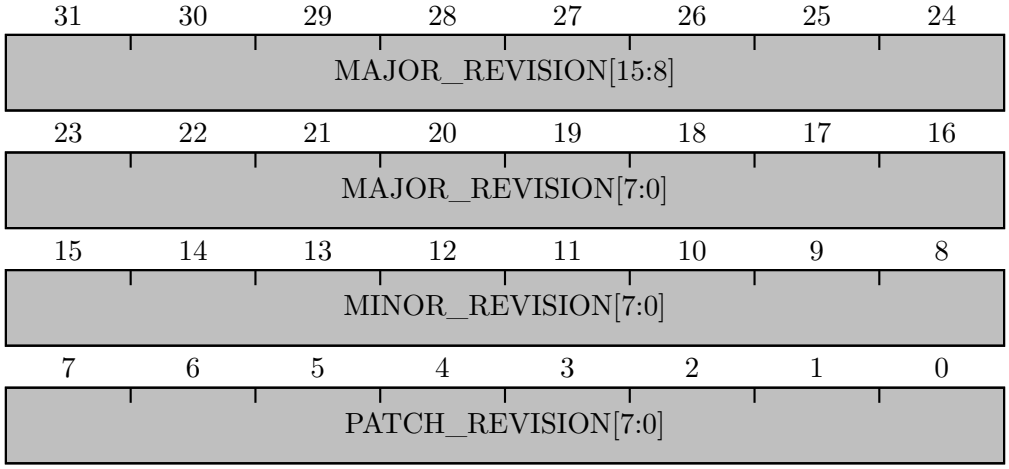
B.4.13 *Registers 11-42 - 0x80 + 4n - IRQ n configuration*



Bitfield	Access	Default value	Description
MODE[1:0]	R/W	'b00	The sensitivity mode of this interrupt. **'b00**: High level. **'b01**: Low level. **'b10**: Rising Edge. **'b11**: Falling Edge.
PEND	R/W	'b0	This bits is a mirror of the corresponding IRQ_PEND bit in the IRQ pending register (appendix B.4.5)
EN	R/W	'b0	This bits is a mirror of the corresponding IRQ_EN bit in the IRQ enable register (appendix B.4.2)
DEST[5:0]	R/W	'b000000	The destination where requests from this IRQ line will be routed to. **'b000000**: PCIe MSI vector 0 (if available). **'b000001**: PCIe MSI vector 1 (if available). **...** **'b011111**: PCIe MSI vector 31 (if available). **'b100000**: CPU IRQ line 0. **'b100001**: CPU IRQ line 1. **...** **'b111111**: CPU IRQ line 31.

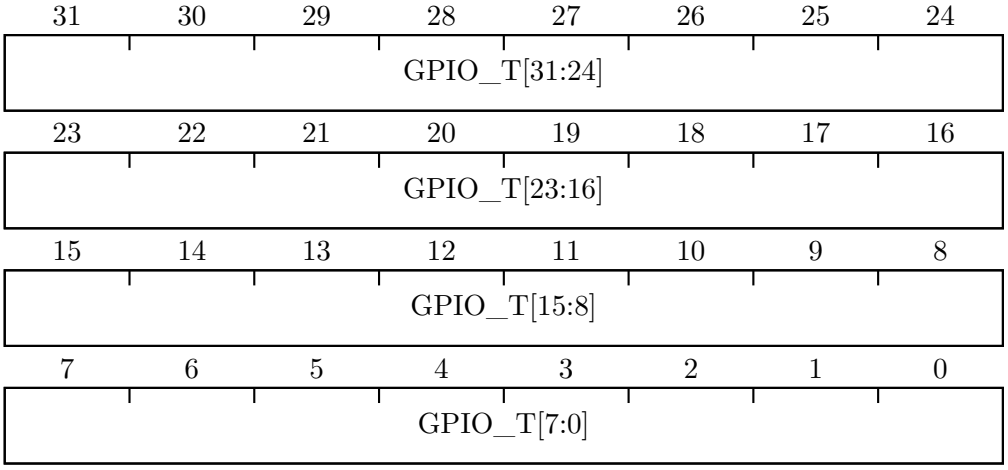
B.5 AXI GPIO IP REGISTER DOCUMENTATION

B.5.1 *Register 0 - 0x00 - IP Version*



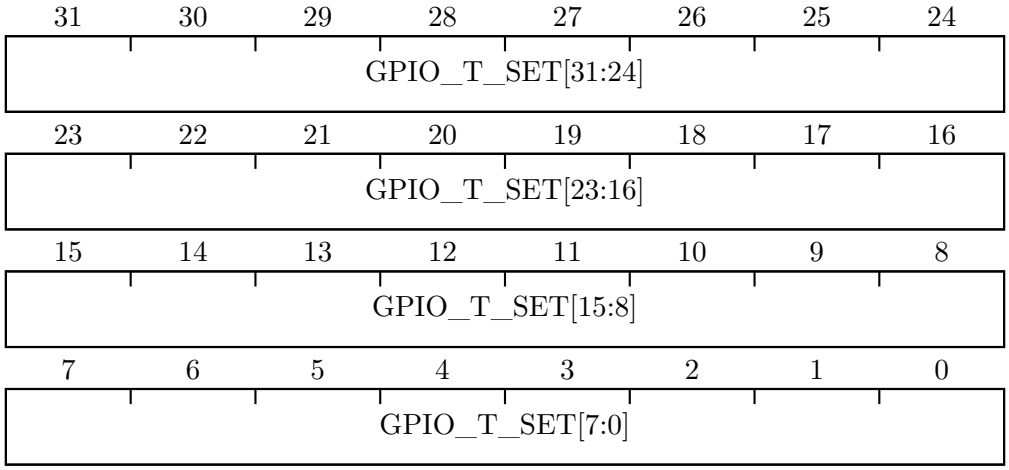
Bitfield	Access	Default value	Description
MAJOR_REVISION[15:0]	R	'h0001	IP module major revision v[**MAJ**].[MIN].[PATCH]
MINOR_REVISION[7:0]	R	'h00	IP module minor revision v[MAJ].[**MIN**].[PATCH]
PATCH_REVISION[7:0]	R	'h00	IP module patch revision v[MAJ].[MIN].[**PATCH**]

B.5.2 *Register 1 - 0x04 - GPIO Direction*



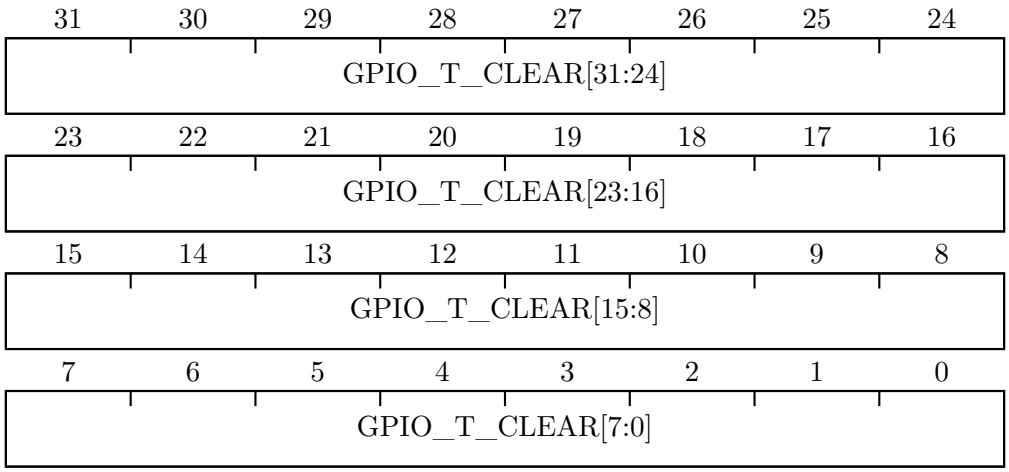
Bitfield	Access	Default value	Description
GPIO_T[31:0]	R/W	'hFFFFFFFF	These bits control the data direction of each GPIO. Each bit controls a single GPIO. A set bit ('b1) configures the corresponding GPIO as an input, while a clear bit ('b0) configures it as an output.

B.5.3 *Register 2 - 0x08 - GPIO Direction (W1TS)*



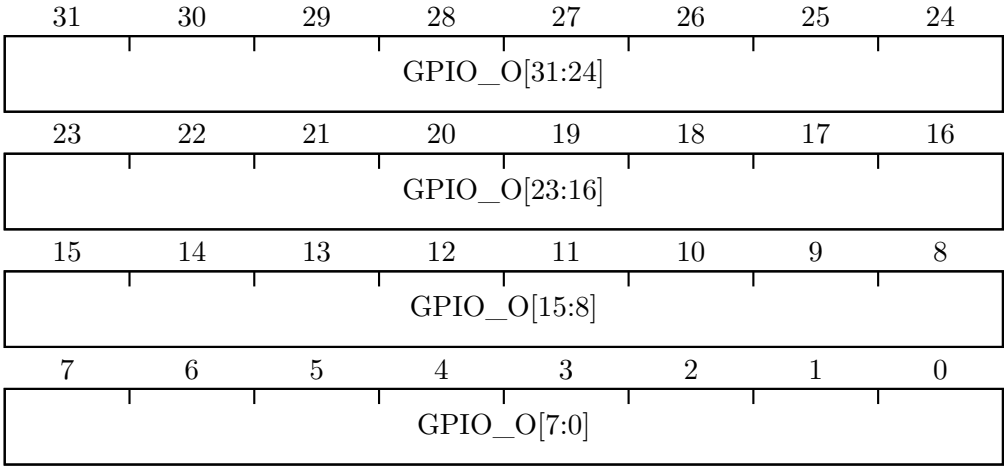
Bitfield	Access	Default value	Description
GPIO_T_SET[31:0]	R/W1	'hFFFFFFFF	Writing one to any of these bits sets the corresponding GPIO_T bit. Writing zero has no effect. The read data is equal to the GPIO direction register (appendix B.5.2).

B.5.4 *Register 3 - 0x0C - GPIO Direction (W1TC)*



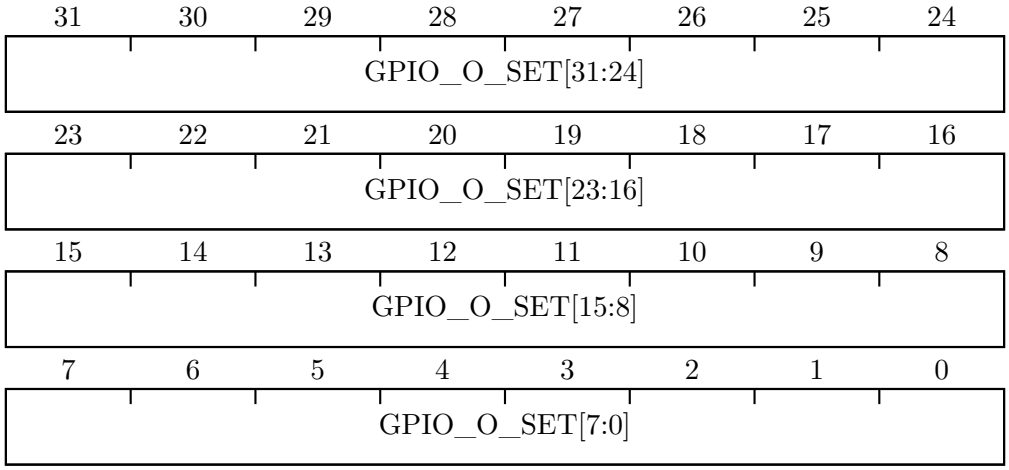
Bitfield	Access	Default value	Description
GPIO_T_CLEAR[31:0]	R/W1	'hFFFFFFFF	Writing one to any of these bits clears the corresponding GPIO_T bit. Writing zero has no effect. The read data is equal to the GPIO direction register (appendix B.5.2).

B.5.5 *Register 4 - 0x10 - GPIO Output Data*



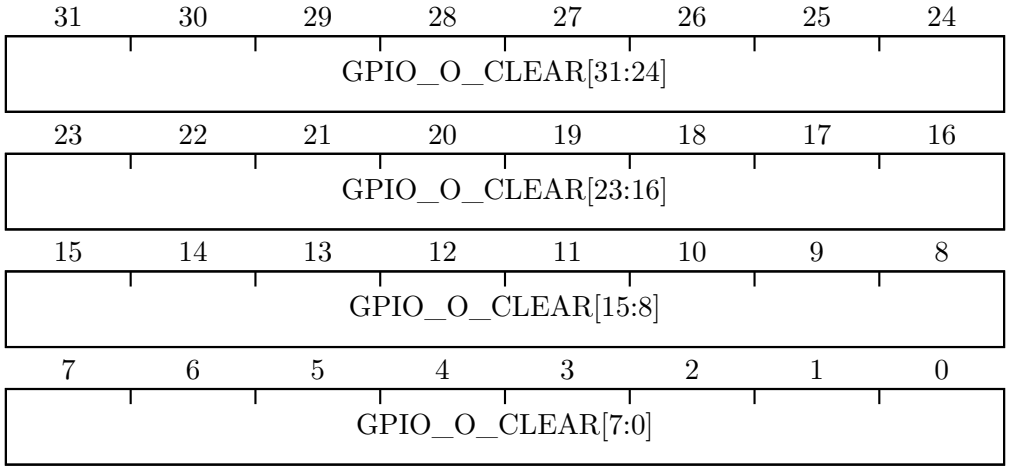
Bitfield	Access	Default value	Description
GPIO_O[31:0]	R/W	'h00000000	These bits control the data present at each GPIO when configured as an output. Each bit controls a single GPIO. A set bit ('b1) configures the corresponding GPIO to output a high value, while a clear bit ('b0) configures it to output a low value.

B.5.6 *Register 5 - 0x14 - GPIO Output Data (W1TS)*



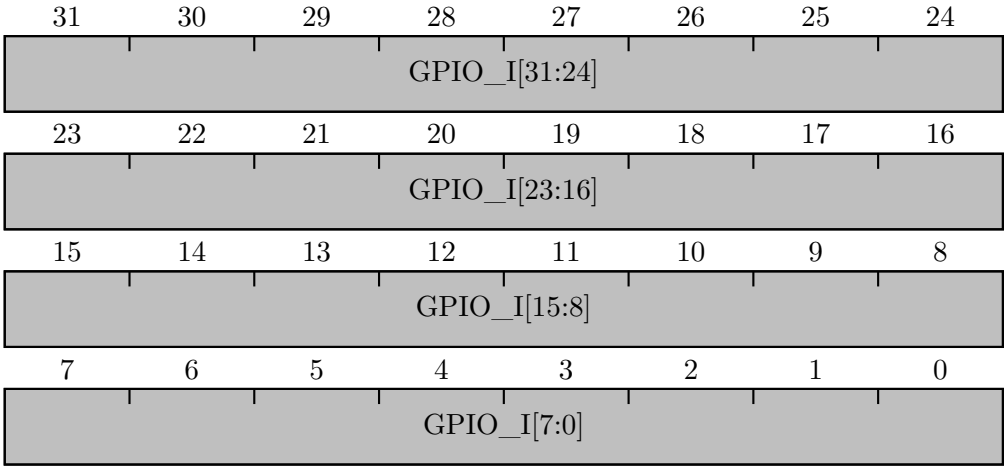
Bitfield	Access	Default value	Description
GPIO_O_SET[31:0]	R/W1	'h00000000	Writing one to any of these bits sets the corresponding GPIO_O bit. Writing zero has no effect. The read data is equal to the GPIO output data register (appendix B.5.5).

B.5.7 *Register 6 - 0x18 - GPIO Output Data (W1TC)*



Bitfield	Access	Default value	Description
GPIO_O_CLEAR[31:0]	R/W1	'h00000000	Writing one to any of these bits clears the corresponding GPIO_O bit. Writing zero has no effect. The read data is equal to the GPIO output data register (appendix B.5.5).

B.5.8 *Register 7 - 0x1C - GPIO Input Data*



Bitfield	Access	Default value	Description
GPIO_I[31:0]	R	'hXXXXXXXX	These bits reflect the value present at each GPIO input, regardless of the direction of the GPIO. A set bit ('b1) indicates a high value present at the GPIO's input, while a clear bit ('b0) indicates a low value.

DECLARATION

I **João Gonçalo Cruz Silva (2222901@my.ipleiria.pt)**, declare, under commitment of honour, that all the work presented in this document, with title ***Millimeter Wave Software Defined Radio***, is original, and has been developed by me, under supervision of **Prof. Doutor Luís Miguel Moreira Mendes (lmendes@ipleiria.pt)** and **Prof. Doutor João Caldinhas Vaz (joaovaz@tecnico.ulisboa.pt)**.

Leiria, March 2025

Assinado por: **João Gonçalo Cruz Silva**
Num. de Identificação: 13721488
Data: 2025.03.29 20:43:28 +0000

CHAVE MÓVEL

João Gonçalo Cruz Silva