FACULTY OF ENGINEERING AND ARCHITECTURE

Hardware Efficient Designs for Deterministic and Low Latency Wireless Communication System Using Software-Defined Radio

Muhammad Aslam

Doctoral dissertation submitted to obtain the academic degree of
Doctor of Electrical Engineering

Supervisors
Prof. Ingrid Moerman, PhD - Xianjun Jiao, PhD
Department of Information Technology
Faculty of Engineering and Architecture, Ghent University

September 2022

GHENT UNIVERSITY

ISBN 978-94-6355-623-1
NUR 959, 965
Wettelijk depot: D/2022/10.500/64

Members of the Examination Board

Chair

Prof. Em. Hendrik Van Landeghem, PhD, Ghent University

Other members entitled to vote

Prof. Gerhard Fettweis, PhD, Technische Universität Dresden, Germany
Prof. Jeroen Hoebeke, PhD, Ghent University
Wei Liu, PhD, Ghent University
Prof. Sofie Pollin, PhD, KU Leuven
Prof. Lieven Tytgat, PhD, Ghent University

Supervisors

Prof. Ingrid Moerman, PhD, Ghent University
Xianjun Jiao, PhD, Ghent University

Acknowledgment

Undoubtedly, the completion of this PhD dissertation was possible from the help and support of various people. First and foremost, I would like to thank and express my deepest gratitude to God Almighty who gave me the knowledge, the strength, his blessing, and the opportunity to conclude this achievement satisfactorily.

Second of all, I would like to extend my gratitude to my promoters, Prof. Dr. Ingrid Moerman and Dr. Xianjun Jiao for proving me the opportunity to study in Ghent University and especially to be a part of the IDLab – an imec research group at Ghent University. During this period, they have been supportive both academically and emotionally through the rough road to finish this thesis. It has been great privilage and a pleasure working with them. Especially, Prof. Dr. Ingrid Moerman, who is so knowledgeable and yet caring. Her words always inspire me and keep me motivated. It was impossible for me to finish this milestone without her kind and patient instructions. Thank you!

Thirdly, I would also like to give special thanks to my office colleague Dr. Wei Liu for her contributions for proof-reading my paper-related writings and my PhD thesis, as feedback is indispensable to successfully complete a PhD. I would also like to thank Prof. Dr. Jeroen Hoebeke and Dr. Adnan Shahid for all insightful comments and the fruitful discussions.

I want to thank the examination board: Prof. Dr. Hendrik Van Landeghem, Prof. Dr. Gerhard Fettweis, Prof. Dr. Lieven Tytgat, Prof. Dr. Sofie Pollin, Prof. Dr. Jeroen Hoebeke, and Dr. Wei Liu for accepting my request to be a part of the thesis committee, reading my PhD thesis and giving constructive feedback, given their unyielding work schedule.

Most of the research work presented in this book has been done during ORCA project and FWO SBO S003921N VERI-END.com project, which are funded by European Union's Horizon 2020 research and innovation programme, and Flemish Government, respectively. I would like to specially thank Prof. Dr. Ingrid Moerman and Prof. Dr. Jeroen Hoebeke for providing me the chance to work on these interesting projects.

I would like to thank all the colleagues in the IDLab, specifically, Dr. Jetmir Haxhibeqiri, Dr. Spilios Giannoulis, Dr. Irfan Jabandzic, Dr. Michael T. Mehari, Dr. Vasilis Maglogiannis, Merkebu Tekaw Girmay, and Thijs Havinga for their cooperation at work and for making this journey with so much fun.

Finally and most importantly, I would like to express my deepest gratitude to my mother, Bashirah beghum, my father, Muhammad Shafi, my life partner, Rabia Rabnawaz, and my siblings for their everlasting and unconditional support and

love on my way.

Gent, September 2022
Muhammad Aslam

Table of Contents

List of Figures

List of Tables

List of Acronyms

0-9

2G 2nd Generation

3G 3rd Generation

3GPP 3rd Generation Partnership Project

4G 4th Generation

5G 5th Generation

A

AP Access Point

ACK Acknowledgment

ADC Analog-to-Digital Converter

ASIC Application Specific Integrated Chip

AMD Advanced Micro Devices

ARM Advanced RISC Machines

AXI Advanced eXtensible Interface

AR/VR Augmented and Virtual Reality

AP SoC All Programmable System on Chip

AGC Automatic Gain Controller

ABC Adder Based Clock

API Application Programming Interface

AGV Automated Guided Vehicle

AMR Autonomous Mobile Robot

B

BW Bandwidth
BLE Bluetooth Low Energy
BBPU Baseband Processing Unit
BRAM Block Random Access Memory
BCIM Bit Coded and Interleaved Modulation
BC Boundary Clock
BSS Basic Service Set
BB Baseband
BBPLL BaseBand Phase Locked Loop

C

cMTC critical Machine Type Communication
CDMA Code Division Multiple Access
CIoT Consumer IoT
CSMA Carrier Sense Multiple Access
CPU Central Processing Unit
CMCVT Concurrent Multi-Channel Virtual Transceiver
CMC-TRX Conventional Multi-Channel Transceiver
CS-FSM Context Switching Finite State Machine
CRC Cyclic Redundancy Check
CMCT Conventional Multi-Channel Transmitter
CMCR Conventional Multi-Channel Receiver
CS Clock Synchronization
CS-BRAM Context Switching Block Random Access Memory
CI Carrier Interferometry
CFO Carrier Frequency Offset
CDF Cumulative Distribution Function
CT Convergence Time
CSMA/CA Carrier Sense Multiple Access/Collision Avoidance

D

DSP Digital Signal Processor
DL Down Link
DSSS Direct Sequence Spread Spectrum
DARPA Defense Advanced Research Projects Agency
DAC Digital-to-Analog Converter
D2D Device-to-Device
DDC Digital Down Converter
DUC Digital Up Converter
HDL Hardware Descriptive Language
DMA Direct Memory Access
DRAM Distributed RAM
DBr Decibel relative

E

ETSI European Telecommunications Standards Institute
ENSM ENable State Machine
ER Extended Range
EVM Error Vector Magnitude
E2E End-to-End
eMBB enhanced Mobile Broadband

F

FPGA Field Programmable Gate Arrays
FHSS Frequency Hopping Spread Spectrum
FDD Frequency Division Duplex
FDMA Frequency Division Multiple Access
FSM Finite State Machine
FD Full-Duplex

FE Front-End
FIR Finite Impulse Response
FF Flip-Flop
FIR Finite Impulse Response
FFT Fast Fourier Transform
FD-MUX Frequency Domain Multiplexer
FIFO First-in First-out
FTM Fine Timing Measurement

G

GPP General Purpose Processor
GPU Graphics Processing Unit
GSM Global System for Mobile Communications
GW Gateway
GMACS Giga Multiply-Accumulates per Second
GM Grand Master
GI Guard Interval
GO Group Orthogonal
GPIO General Purpose Input and Output

H

HW Hardware
HR Hardware Radio
HA Hardware Accelerator
H2H Human-to-Human
HD Hardware Duplication
HV Hardware Virtualization
HE High Efficiency
HE-SIGB HE Signal B
HE-SIGA HE Signal A
HE-STF HE Short Training Field

HE-LTF HE Long Training Field
HB Half-Band

I

ISR Ideal Software Radio
ISI Inter Symbol Interference
IoT Internet of Things
IEEE Institute of Electrical and Electronics Engineers
IIoT Industrial IoT
ISM Industrial, Scientific and Medical
IQ In-phase and Quadrature phase
ILA Integrated Logic Analyzer
IF Intermediate Frequency
IoE Internet of Everything
IFFT Inverse Fast Fourier Transform
IH Interrupt Handler
IP Internet Address
IWSN Industrial Wireless Sensor Network

K

kbps kilobit per second

L

LR-WPAN Low Rate WPAN
LAN Local Area Network
LNA Low Noise Amplifier
LTE Long Term Evolution
LPWAN Low Power Wide-Area Network
LoRa Long Range
LUT Look Up Table

L-STF Legacy Short Training Field
L-LTF Legacy Long Training Field
L-SIG Legacy Signal
LBT Listen Before Talk
LO Local Oscillator

M

MU Multi User
MIMO Multiple Input Multiple Output
MAC Media Access Control
MPDU MAC Payload Data Unit
mMTC massive Machine Type Communication
MAN Metropolitan Area Network
M2M Machine-to-Machine
MPAP Multi-Protocol Access Point
MCVT Multi-Channel Virtual Transmitter
MCVR Multi-Channel Virtual Receiver
MCS Modulation Coding Scheme
MLME MAC Layer Management Entity

N

NTP Network Time Protocol
NB Narrowband
NR New Radio
NI National Instruments
NIC Network Interface Card
NB Narrow-Band

O

OFDM Orthogonal Frequency Division Multiplexing

OFDMA Orthogonal Frequency Division Multiple Access
OSI Open Systems Interconnection
OS Operating System
O-QPSK Offset Quadrature Phase Shift Keying
OC Ordinary Clock

P

PHY Physical
PC Personal Computer
PCI Peripheral Component Interconnect
PTP Precision Time Protocol
gPTP generalized PTP
PHC PTP Hardware Clock
PSS Power Saving Scheme
PL Programmable Logic
PS Processor System
PLL Phase Locked Loop
PSD Power Spectrum Density
PER Packet Error Rate
PAPR Peak to Average Power Ratio
PPDU Protocol Data Unit
PoC Proof of Concept
P2P Peer-to-Peer
PI Proportional Integral
PLD Programmable Logic Device
PA Power Amplifier
PPS Pulse Per Second

Q

QoS Quality of Service

R

RF Radio Frequency
RX Receiver
Rx Receiving
RU Resource Unit
RF-FE Radio Frequency Front-end
RAM Random Access Memory
RU Resource Unit
RL-SIG Repeated Legacy Signal
RTT Round-Trip Time

S

SCR Software Controlled Radio
SoC System on Chip
SC Subcarrier
SU single User
STA Station
SDR Software Defined Radio
SIFS Short Interframe Spacing
SISO Single Input Single Output
SS Spatial Stream
SW Software
SCC Skew Correction Counter
SCREG Skew Correction Register

T

TRX Transceiver
TX Transmitter
Tx Transmitting

TSN Time Sensitive Networking
TDD Time Division Duplex
TDMA Time Division Multiple Access
TT Turnaround Time
TI Texas Instrument
TSF Timing Synchronization Function
TB Triggered Based
TD Time Domain
TD-MUX Time Domain Multiplexer
TS TimeStamping
TC Transparent Clock
TM Timing Measurement
TSU TimeStamp Unit
TCP Transmission Control Protocol
TAISC Time-Annotated Instruction Set Computer

U

USR Ultimate Software Radio
UL Up Link
UDP User Datagram Protocol
USB Universal Serial Bus
USRP Universal Software Radio Peripheral
URLLC Ultra-Reliable Low Latency Communications
UWB Ultra Wide Band

V

VLAN Virtual Local Area Network
VCO Voltage Controlled Oscillator

W

WBAN Wireless Body Area Network

WPAN	Wireless Personal Area Network
WLAN	Wireless Local Area Network
WMAN	Wireless Metropolitan Area Network
WTSN	Wireless TSN
WWAN	Wireless Wide Area Network
WAN	Wide Area Network
Wi-Fi	Wireless Fidelity
WiMAX	Worldwide Interoperability for Microwave Access
WB-RF	Wideband RF
WIA-FA	Wireless networks for Industrial Automation-Factory Automation
WB	Wide-Band
WSN	Wireless Sensor Network

Samenvatting
– Summary in Dutch –

Tegenwoordig worden draadloze communicatiesystemen gebruikt in tal van toepassingen, variërend van consumententoepassingen zoals het klassieke internet en het internet der dingen (Internet of Things, IoT) tot industriële toepassingen, zoals het industriële internet der dingen (Industrial IoT, IIoT) of Industrie 4.0. Hiertoe is er over de hele wereld een overvloed aan draadloze communicatiestandaarden (of eigen oplossingen op maat) ontwikkeld, die elk voldoen aan de specifieke 'Quality of Service' (QoS) vereisten wat betreft bereik, energieverbruik, kosten, responstijd, betrouwbaarheid, datasnelheid, enzovoort. Deze standaarden evolueren voortdurend door de introductie van nieuwe technieken zoals Orthogonal Frequency Division Multiple Access (OFDMA), Multiple-Input Multiple-Output (MIMO), verbeterde modulatie- en coderingsschema's, een grotere spectrale bandbreedte en uitbreiding met nieuwe frequentiebanden. Enerzijds vergt de realisatie van deze geavanceerde functionaliteit van standaarden complexe hardware-ontwerpen voor de 'baseband' verwerkingseenheid (Baseband Processing Units, BBPUs) van de radiozenders en -ontvangers (Transceivers, TRXs)[1]. Met andere woorden, er zijn meer hardwaremiddelen nodig om dergelijke complexe hardware-ontwerpen te implementeren. Aan de andere kant vragen deze standaarden ook om flexibele, herconfigureerbare en programmeerbare platformen, aangezien er voortdurend nieuwe geavanceerde functies en verbeteringen in deze standaarden worden geïntroduceerd. Derhalve zijn hardware-efficiënte ontwerpen voor de TRX's op flexibele, herconfigureerbare en programmeerbare platformen zeer gegeerd voor de realisatie en validatie date van moderne draadloze communicatiesystemen.

Een software-gedefinieerd radiosysteem (Software-Defined Radio, SDR) is het meest veelbelovende platform voor het ontwerpen van de TRX's van de nieuwste draadloze communicatiesystemen, aangezien de allernieuwste SDR niet alleen flexibel is, maar ook geavanceerde digitale verwerkingsmogelijkheden ondersteunt. Een SDR bestaat hoofdzakelijk uit twee delen: een programmeerbare

[1]Een radiozender en -ontvanger bestaat grofweg uit een BBPU, analoog-naar-digitaal omzetter (Analog to Digital Converter, ADC)/digitaal-naar-analoog omzetter (Digital to Analog Converter, DAC) en radiofrequenti (RF) front-end. Terwijl de BBPU de signaalverwerking uitvoert op basebandniveau in het digitale domein, werkt de RF-front-end in het analoge domein en omvat upconverters en eindversterkers aan de zenderzijde, en lage-ruisversterkers (Low Noise Amplifiers, LNAs) en downcoverters aan de ontvangerzijde. De ADC en DAC kunnen werken op basebandfrequenties of tussenfrequenties en fungeren als een brug tussen de analoge en digitale domeinen.

digitale component en een configureerbare analoge RF front-end. De digitale component, die bestaat uit een aantal digitale BBPUs die zijn geïmplementeerd op programmeerbare apparaten, zoals een digitale signaalverwerker (Digital Signal Processor, DSP), een Field Programmable Gate Array (FPGA) of een centrale verwerkingseenheid (Central Processing Unit, CPU). De herconfigureerbare analoge RF front-end van een SDR omvat simpelweg alles tussen de BBPU en de antenne. De gebruikelijke componenten van een RF front-end zijn een versterker, mixer, filters, DAC en ADC. Dit proefschrift richt zich op hardware-efficiënte ontwerpen met focus op een snelle responstijd en deterministische draadloze communicatiesystemen die gebruik maken van SDR. Meer specifiek worden eerst hardware-efficiënte architecturen voor de complexe hardware-ontwerpen van radio TRX's voorgesteld. Vervolgens worden concrete technieken geïntroduceerd om de prestaties van TRX's gebouwd op een SDR te optimaliseren. Tenslotte wordt een geoptimaliseerde oplossing gegeven voor kloksynchronisatie, die de basis vormt van deterministische netwerken die deels bekabeld en deels draadloos zijn.

In de industriële versie van IoT (of IIoT), gericht op industriële verticale markten als zoals automatisering van productieprocessen, worden specifieke eisen gesteld aan IoT-netwerken. Datasnelheid is niet de enige zorg in dergelijke IoT-netwerken, maar nog belangrijker is een snelle en begrensde responstijd en een betrouwbare datatransfer die vraagt om een kleine variatie in responstijd en een extreem laag verlies van data. In een draadloos IoT (of IIoT) communicatienetwerk is een gateway (GW) of toegangspunt (AP) meestal uitgerust met een enkelvoudige TRX radio-interface (hierna 'enkelvoudige gebruiker' of 'Single User' (SU) TRX genoemd) die slechts kan communiceren met één eindapparaat ('of station (STA)) tegelijkertijd[2]. Aangezien het aantal eindtoestellen die verbonden is met het internet aanzienlijk is toegenomen, heeft een SU TRX gebaseerde GW een nadelige invloed op de prestaties van een IoT-netwerk wat de responstijd betreft wanneer meerdere slimme toestellen tegelijk toegang proberen te krijgen tot de GW. Een manier om de prestaties van de GW te verbeteren is het gebruik van MU TRX, d.w.z. een TRX die meerdere gebruikers tegelijk kan bedienen. Veel draadloze communicatiestandaarden zoals IEEE 802.11ax en de 5de Generatie (5G) ondersteunen MU-communicatie door de introductie van MU-OFDMA en MU-MIMO technologieën. MU-communicatie kan grofweg in twee categorieën worden ingedeeld. In de eerste categorie neemt de capaciteit van de draadloze communicatiestandaard toe met het aantal gebruikers in de MU-communicatie. Voorbeelden van een dergelijke categorie zijn MU-MIMO en simultane communicatie over meerdere kanalen waarbij elk afzonderlijk kanaal aan een andere gebruiker is toegewezen. In de tweede categorie blijft de capaciteit van de draadloze communicatiestandaard vast, ongeacht het aantal gebruikers in MU-communicatie (bv. MU-OFDMA in een kanaal met vooraf gedefinieerde bandbreedte). Ongeacht de categorie stelt de vraag naar MU-functionaliteit een nieuwe uitdaging voor het ontwerp van draadloze basebandchips. Bestaande MU TRX's berusten

[2]In een SU TRX ontvanger kan een gebruiker evenwel communiceren met meerdere gebruikers, maar niet gelijktijdig.

op een duplicatie-aanpak waarbij afzonderlijke hardware per gebruiker wordt gebruikt. De hardware voetafdruk van een dergelijke benadering neemt dus proportioneel toe met het aantal gebruikers dat gelijktijdig door de MU TRX bediend wordt. Daarom wordt, als een eerste bijdrage van dit doctoraat, een nieuw hardware efficiënt ontwerp voor een MU TRX geïntroduceerd. In tegenstelling tot de duplicatie-aanpak, heeft het voorgestelde ontwerp voor de MU TRX een vergelijkbare hardware footprint als een SU TRX, ongeacht het aantal gebruikers die bediend worden, dankzij de hardware virtualisatie techniek. De toepasbaarheid van het ontwerp wordt eerst gevalideerd voor een multi-kanaal TRX conform aan de Institute of Electrical and Electronics Engineers (IEEE) 802.15.4 standaard en vervolgens voor een MU-OFDMA zender conform aan de IEEE 802.11ax standaard. Voor beide gevallen wordt de werking gevalideerd op een FPGA van een moderne SDR. De prestaties en het hardware-verbruik van hardware-virtualisatie aanpak worden vergeleken met de conventionele duplicatie-aanpak, en er wordt een aanzienlijke besparing op hardware-middelen gerealiseerd.

Draadloze communicatiestandaarden zoals IEEE 802.11 en 5G die in Time Division Duplex (TDD) werken, hebben strikte eisen qua responstijd. Zo moet de IEEE 802.11 standaard bij werking in de 2,4 GHz frequentieband, een ACK ('Acknowledgement') pakket verzenden binnen de 10 microseconden na de succesvolle ontvangst van een datapakket. De op zichzelf staande TDD-werking in 5G New Radio (NR) vereist ook een ACK binnen een wachttijd van enkele microseconden. Los van het feit dat moet worden voldaan aan de strikte timingvereisten van radioprotocollen voor toegangscontrole tot het draadloze medium (MAC), zoals snelle ontvangstbevestigingen, is een snelle TT 'Turnaround Time', d.w.z. de tijd die een radio nodig heeft om van de ontvangstmodus over te schakelen op de zendmodus of omgekeerd), ook van kritisch belang om de kans op botsingen bij gebruik van een MAC met "Listen Before Talk"(LBT) te verminderen. Ondanks de hoge digitale verwerkingsmogelijkheden en flexibiliteit die de moderne SDR's kunnen bieden, vormen de trage TT's van hun analoge RF front-ends, typisch in de orde van tientallen microseconden, een ernstige beperking voor het onderzoek en de validatie van toekomstige draadloze communicatiestandaarden op een SDR-platform. Daarom wordt in de tweede bijdrage van dit doctoraatsboek een algemene oplossing geïntroduceerd om de TT op een SDR te verminderen. De nieuwe oplossing, waarbij de BBPU in half duplex modus werkt en het analoge RF front-end in full duplex modus werkt, wordt voorgesteld om de TT op een SDR te verminderen. De voorgestelde oplossing is gevalideerd op de veelgebruikte AD9361 RF front-end. Experimenten tonen aan dat met de voorgestelde oplossing de TT in TDD-modus tot nul kan worden herleid, met een verwaarloosbare impact op het communicatiesysteem qua gevoeligheid van de ontvanger. De impact is gemeten voor een in-huis ontwikkelde TRX conform aan de IEEE 802.15.4 standaard. Het concept is generiek en kan worden toegepast op elke hoogwaardige draadloze communicatietechnologie, al dan niet conform aan een standaard.

Naast een snelle responstijd is determinisme (d.w.z. gegarandeerd pakkettransport met een begrensde responstijd, kleine variatie van de responstijd voor pakketten en laag pakketverlies) een eerste vereiste voor tijdkritische industriële toepas-

singen zoals mobiele robots, gesloten-lus regelsystemen in productieprocessen. Om deterministische draadloze connectiviteit te bereiken, moeten draadloze transmissies nauwkeurig gepland worden, bijvoorbeeld door gebruik te maken van een Time Division Multiple Access (TDMA) schema, wat nauwkeurige kloksynchronisatiemechanismen vereist tussen apparaten die deelnemen in het deterministische netwerk. Time Sensitive Networking (TSN) is een set van IEEE 802 standaarden die verantwoordelijk zijn voor het leveren van deterministische connectiviteit over bekabelde netwerken. Er is een groeiende belangstelling voor de draadloze uitbreiding van TSN, voornamelijk ingegeven door de recente vooruitgang op het gebied van draadloze lokale netwerken (WLAN) en 5G-standaarden. TSN is gebaseerd op het principe dat alle knooppunten in een gedistribueerd netwerk dezelfde notie van tijd delen, wat mogelijk is met kloksynchronisatieprotocollen. PTP, een recent kloksynchronisatieprotocol is in eerste instantie ontworpen voor bekabelde netwerken en heeft recent aan aandacht gewonnen in de draadloze gemeenschap, wegens de toenemende belangstelling van in TSN voor het vervangen van bekabelde connectiviteit door draadloze connectiviteit voor meer dynamische gebruiksscenario's. Om PTP te realiseren moet het onderliggende netwerkapparaat in staat zijn om nauwkeurige tijdstempels te verstrekken aan pakketten die de netwerkinterface binnenkomen en verlaten. De tijdstempels kunnen in software (ook bekend als stuurprogramma of firmware) of in hardware (d.w.z. met behulp van een specifieke hardwareklok dichter bij de fysieke laag) worden vastgelegd. Bestaande PTP ontwerpen voor draadloze netwerken bevatten ofwel softwaretijdstempels die een lage nauwkeurigheid van de kloksynchronisatie opleveren, ofwel hardwaretijdstempels die een betere synchronisatienauwkeurigheid bieden, ten koste van protocolcompatibiliteit en waarvoor een aanzienlijke hoeveelheid hardwaremiddelen nodig zijn. Daarom wordt, als derde bijdrage van dit doctoraatsboek, een nieuwe aanpak geïntroduceerd en experimenteel gevalideerd om PTP te implementeren voor de IEEE 802.11 WLAN standaard op een SDR platform. In plaats van een specifieke hardware klok te gebruiken, maakt de oplossing gebruik van de Timing Synchronization Function (TSF) klok, een bestaande klok in de IEEE 802.11 standaard voor synchronisatie tussen access point en WLAN stations. Een nauwkeurigheid van synchronisatie in de orde van microseconden is aangetoond zowel binnen een single-hop WLAN als in netwerken die deels bekabeld en deels draadloos zijn.

Van hardware-efficiënte ontwerpen voor MU communicatie voor meerdere draadloze standaarden, tot zeer nauwkeurige kloksynchronisatie in een draadloos netwerk, biedt dit proefschrift verschillende hardware-efficiënte digitale hardware oplossingen voor een deterministisch draadloos netwerk met snelle responstijd, die dienen als belangrijke bouwstenen naar volgende generatie draadloze netwerken gericht op tijdsgevoelige industriële toepassingen.

Summary

Nowadays, wireless communication systems have been used in numerous applications ranging from consumer-oriented applications such as Internet and Consumer Internet of Things (IoT) to industrial applications, e.g., Industrial IoT (IIoT), Industry 4.0. To this end, a plethora of wireless communication standards (or proprietary solutions) have been developed across the world with each addressing specific Quality of Service (QoS) requirements in terms of range, energy consumption, cost, latency, reliability, data rate and so on. These standards are constantly evolving by introducing new techniques such as OFDMA, MIMO, enhanced modulation and coding schemes, increasing channel bandwidth, and adding new frequency bands. On the one hand, realization of these advanced features of the standards require complex Hardware (HW) designs for Baseband Processing Units (BBPUs) of the corresponding radio TRXs[3]. In other words, more HW resources are required to implement such complex HW designs. On the other hand, these standards also demand a flexible, reconfigurable and programmable platform, as new advanced features and improvements are gradually introduced in these standards. Thus, hardware efficient designs for the radio TRXs on flexible, reconfigurable and programmable platforms are highly desirable for prototyping and validation of modern wireless communication systems.

Software Defined Radio (SDR) is the most promising platform for designing the radio TRXs of modern wireless communication systems, as a state-of-the-art SDR is not only flexible, but also supports advanced digital processing capabilities. An SDR is primarily composed of two parts: a programmable digital component and a reconfigurable analog Radio Frequency (RF) front-end. The digital component consists of a number of digital BBPUs implemented on programmable devices, such as a Digital Signal Processor (DSP), an Field Programmable Gate Arrays (FPGA), or a Central Processing Unit (CPU). The reconfigurable analog RF front-end of an SDR is simply everything between the BBPUs and the antenna. The general components pertaining to an RF front-end are amplifier, mixer, filters, Digital-to-Analog Converter (DAC), and Analog-to-Digital Converter (ADC). This PhD dissertation focuses on HW efficient designs for low latency and de-

[3]A radio TRX broadly consists of BBPU, Analog-to-Digital Converter (ADC)/Digital-to-Analog Converter (DAC), and Radio Frequency (RF) front-end. While the BBPU performs the signal processing at the baseband level in digital domain, the the RF front-end operates in analog domain and comprises upconverters and power amplifiers on the Transmitter (TX) side, and Low Noise Amplifiers (LNAs) and downconverters on the Receiver (RX) side. The ADC and DAC, which operates at baseband frequencies or intermediate frequencies, act as a bridge between analog and digital domains.

terministic wireless communication systems using SDR. More specifically, first, hardware efficient architectures for the complex HW designs of radio TRXs are presented. Then, concrete techniques to optimize the performance of TRXs built on an SDR are introduced. Finally, an optimized solution for clock synchronization, which is the foundation of deterministic wired/wireless networks, is given.

In the industrial version of IoT (or IIoT) targeting industrial verticals such as automation of manufacturing processes imposes specific requirements on IoT network. Data rate is not the only concern in such IoT networks, but even more important is low and bounded latency and reliable delivery calling for low delay variation and extremely low loss. In an IoT (or IIoT) wireless communication network, a Gateway (GW) (or Access Point (AP)) mostly equipped with a single TRX radio interface (hereinafter referred to as a single User (SU) TRX) is able to communicate with a single end-device (or Station (STA)) at a time[4]. Since the number of end-devices connected with the Internet in an IoT network has proliferated significantly, A SU TRX based GW adversely affects the performance (in particular in terms of latency) of an IoT network when multiple smart devices try to access the GW at the same time. One way to improve the performance of the GW is to use MU TRX, meaning a TRX that can serve multiple users at the same time. Many wireless communication standards such as IEEE 802.11ax and 5th Generation (5G) support MU communication by introducing MU-OFDMA and MU-MIMO technologies. The MU communication can be broadly classified into two categories. In the first category, the capacity of the wireless communication standard increases with the number of users in MU communication. Examples of such category are MU-MIMO and concurrent multi-channel communication with each channel dedicated to a user. In the second category, the capacity of the wireless communication standard remains fixed irrespective to the number of users in MU communication (e.g., MU-OFDMA in a predefined fixed bandwidth). No matter which category, the demand for MU functionality poses new challenges on the baseband wireless chip design. Existing MU TRXs rely on the duplication approach wherein a dedicated hardware is utilized per user. The hardware footprint of such an approach increases proportionally with the number of users simultaneously served in the MU TRX. Therefore, as a first contribution of this PhD book, a novel hardware efficient design for an MU TRX is introduced. Unlike the duplication approach, the proposed design for the MU TRX has comparable hardware footprint of a SU TRX regardless of the number of users being served, thanks to the hardware virtualization technique. The applicability of the design is first validated for an Institute of Electrical and Electronics Engineers (IEEE) 802.15.4 compliant concurrent multi-channel TRX and then for an IEEE 802.11ax compliant MU-OFDMA transmitter. For both cases, the validation is achieved on an FPGA of a modern SDR. The performance and hardware consumption of the hardware virtualization based approach is compared against the conventional duplication approach, and a significant reduction in hardware resources is achieved.

[4]Although the receiver in SU TRX is capable to communicate with multiple users but not concurrently.

The wireless communication standards such as IEEE 802.11 and 5G operating in Time Division Duplex (TDD) have strict latency requirements. For instance, IEEE 802.11 standard needs to send an Acknowledgment (ACK) packet within 10 microseconds (when operating in the 2.4 GHz frequency band) after the successful reception of a data packet. The self-contained TDD operation in 5G New Radio (NR) also needs an ACK within a few microseconds range. Apart from meeting stringent timing requirements of Media Access Control (MAC) protocols such as fast acknowledgements, a low Turnaround Time (TT), being the time required by a radio to change from Receiving (Rx) mode to Transmitting (Tx) mode or vice versa), is also critical to reduce the probability of collisions in a Listen Before Talk (LBT) like MAC operation. Despite the high digital processing capabilities and flexibility the modern SDRs can offer, the slow TTs of their analog RF front-ends, typically in the order of tens of microseconds, seriously constrain research and validation of future wireless communication standards on SDR platforms. Therefore, in the second contribution of this PhD book, a generalized solution to reduce the TT on a modern SDR is introduced. The novel solution, in which the BBPU operates in half-duplex mode and the analog RF front-end operates in full-duplex mode, is presented to reduce the TT on an SDR. The proposed solution is validated on the widely adopted AD9361 RF front-end. Experiments unveil that with the proposed solution the TT in TDD mode can be reduced to zero , with negligible impact on the communication system in terms of receiver sensitivity. The impact is measured for an in-house IEEE 802.15.4 compliant TRX. The concept is generic and could be applied to any high-performance wireless communication standard (or proprietary solution).

In addition to low latency, determinism (i.e., guaranteed packet transport within bounded latency, low packet delay variation, and low packet loss) is a prerequisite for time-critical industrial applications such as mobile robots, close-loop control in manufacturing processes. To achieve deterministic wireless connectivity, wireless transmissions need to be precisely scheduled, for instance using a Time Division Multiple Access (TDMA) scheme, which demands an accurate clock synchronisation mechanisms between devices participating in the deterministic network. Time Sensitive Networking (TSN) is a set of IEEE 802 standards responsible for providing deterministic connectivity over wired networks. There is a growing interest in the wireless extension of TSN, mainly motivated by recent advances in Wireless Local Area Network (WLAN) and 5G standards. The TSN is based on the principle that all the nodes in a distributed network share the same notion of time, which is possible using clock synchronization protocols. PTP, a state-of-the-art clock synchronization protocol is primarily designed for wired networks and has recently gained attention in the wireless community, due the increasing interest in TSN for replacing wired connectivity by wireless connectivity for more dynamic use scenarios. To realize PTP, the underlying network device must have the capability to provide accurate timestamps to packets entering and leaving the network interface. The timestamps can be captured in software (i.e., in device drivers or at a higher layer of network stack without hardware assistance) or in hardware (i.e., at or closer to physical layer assisted by a dedicated hardware clock). Existing

wireless network based PTP designs either incorporate software timestamping delivering poor clock synchronization accuracy, or hardware timestamping providing better synchronization accuracy at the cost of losing protocol compatibility and a significant amount of hardware overhead. Therefore, as a the third contribution of this PhD book, a new SDR based approach to implement PTP is introduced and experimentally validated for the IEEE 802.11 WLAN standard. Instead of using a dedicated hardware clock, the solution utilizes the Timing Synchronization Function (TSF) clock, an existing clock in IEEE802.11 standard for synchronization between access point and WLAN stations. Microsecond level synchronization accuracy has been demonstrated within a single-hop WLAN as well as across wired-wireless networks.

Starting from hardware efficient designs for MU communication for multiple wireless standards, to high precision clock synchronization in a wireless network, this PhD dissertation provides several hardware efficient digital hardware solutions for a low latency and deterministic wireless network, which serve as important building stones towards next generation wireless networks targeting time-sensitive industrial applications.

1 Introduction

"If you want something new, you have to stop doing something old."

–Peter Drucker (1909 - 2005)

This chapter begins with the introducing the history, standards, applications and the evolution of wireless communication systems. It is followed by description of an Software Defined Radio (SDR), the radio platform that has been used for prototyping and validating new concepts explored in this dissertation. Next, the motivations and challenges addressed in this PhD research are elaborated. Finally, the main contributions and outline of the dissertation together with the lists of publications authored during this research period are detailed.

1.1 Wireless Communications

This section first briefly describes the history of wireless communication. Next, it introduces the main wireless communication standards. Then, the existing and emerging applications of wireless communication are presented. Lastly, the evolution of communication standards is mentioned.

1.1.1 The History of Wireless Communications

The first wireless communication system is developed in pre-historic period. The earliest wireless connections were achieved by transmitting visual signals in line-of-sight conditions, including smoke, or flashing a mirror. The origin of radio

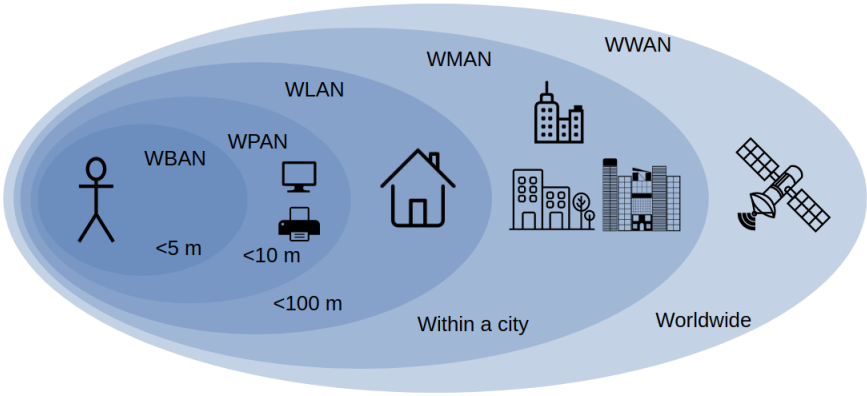


Figure 1.1: Classification of wireless communication networks.

technology began when James C. Maxwell theoretically proved the existence of electromagnetic waves in the 1860s. The presence of electromagnetic waves was experimentally proven by Heinrich R. Hertz when he transmitted the first electromagnetic wave in the late 1880s. In 1895, Guglielmo Marconi demonstrated the first successful transmission and reception of a Morse code over a radio signal, which is considered as the birth of modern radio communication system. A first speech signal was transmitted over the radio waves in 1900 by Fessenden, and six years later he made the first public radio broadcast [1].

After that, wireless communication systems were mainly advanced for military uses. In 1980s, Defense Advanced Research Projects Agency (DARPA) invested significant resources to develop the wireless communication networks used in the battlefields. The communication network grouped the bits into packets, which were are sent over radio signals as individual information units. In early 1990s, the wireless communication was also expanded for commercial purposes to provide wide area wireless data services including email, web browsing, file transfer at a data rate of 20 Kbps [1]. Subsequently, the introduction of cellular telephones and wireless local area networks has marked the dawn of the wireless data revolution.

1.1.2 Wireless Communication Standards

Since the introduction of the first wireless communication system, wireless technology has been continuously evolving. Releases of wireless standards (or protocols) describe the subsequent innovations of wireless technology evolutions. Different wireless communication standards have been developed across the world, based on various requirements, including, but not limited to, range, energy consumption, cost, latency, reliability, and data rate. As illustrated in Fig. 1.1, these standards can be classified into five individual categories, depending on the area

of application and transmission range.

Wireless Body Area Network (WBAN) is devised to work on human body (but not limited to human body) and its surrounding with a maximum range up to 5 meters [2]. The WBAN is based on Institute of Electrical and Electronics Engineers (IEEE) 802.15.6. Wireless Personal Area Network (WPAN) is a type of network wherein the devices can communicate within an area of about 10 meters. Based on data rate, WPAN is sub-categorized into Low Rate WPAN (LR-WPAN), which includes IEEE 802.15.4, and WPAN, which includes IEEE 802.15.1 (commercially known as Bluetooth), IEEE 802.15.3 Ultra Wide Band (UWB), and IEEE 802.11ah (also known as HaLow) [3–6].

A Wireless Local Area Network (WLAN) is used in indoor spaces such as home, office building, school or computer laboratory and provides wireless access in a typical range of up to 100 meters. WLAN is mainly based on the IEEE 802.11 family of standards (i.e., IEEE 802.11a/b/g/n/ac/ax), commercially known as Wireless Fidelity (Wi-Fi) [7, 8].

Wireless Metropolitan Area Network (WMAN) allows the users to establish wireless connections between different locations within a metropolitan areas such as a city. In the WMAN, the most frequently used standard is IEEE 802.6, which is also known as Worldwide Interoperability for Microwave Access (WiMAX) [9]. The WMAN cover areas which range from 100 meters to several tons of kilometers.

Wireless Wide Area Network (WWAN) extends beyond 50 kilometers and cover large areas, such as cities or countries, via satellite or cellular systems. Depending on the power consumption and data rate, The WWAN can be further categorized into Low Power Wide-Area Network (LPWAN) and WWAN. The LPWANs include Long Range (LoRa) [10], and Sigfox [11] consume low power and offer low data rate over larger areas. WWANs are mainly based on the mobile communication standards developed by the 3rd Generation Partnership Project (3GPP) [12] and satellite communication standards maintained by European Telecommunications Standards Institute (ETSI) [13].

1.1.3 Applications of Wireless Communication

Connection between human beings is one of the most important applications of wireless communication available on the Internet. The Internet is a global network that enables humans from all over the world to share information and communicate with one another (also known as the internet of people). Some examples of human-centric applications over Internet are chatting over smart phones, laptops, or interacting with each other in online gaming. The continuous evolution of wireless communication standards expands the scope of Internet from Human-to-Human (H2H) communication towards Machine-to-Machine (M2M) communica-

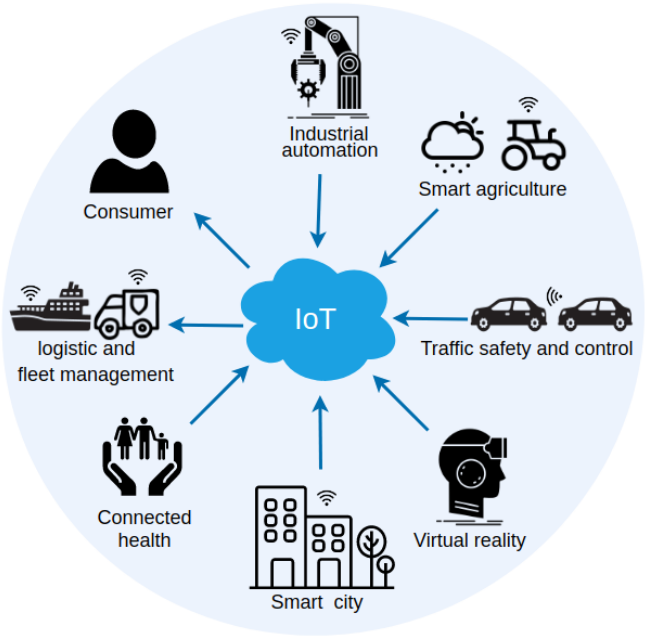


Figure 1.2: Internet of Things (IoT) application domains and markets.

tion. M2M communication over Internet is known as Internet of Things (IoT) [14].

IoT, which was first coined in 1999, in a broad sense encompasses all the things (or devices) connected to the Internet. Unlike the traditional internet access where internet is used for the communication between humans, IoT enables devices (or machines) to collect and transfer data over the network without human intervention. Although a network could be wired or wireless, the latter is preferred as it offers flexible and cost-effective connections. IoT has slowly and steadily captured almost every aspect of human lives. The number of connected IoT devices, which was 14.6 billions in 2021, is estimated to surpass 30 billions in 2027 [15]. This is due to the rapid rise of IoT applications in different sectors and market verticals, as shown in Fig. 1.2, ranging from manufacturing, transportation, agriculture, smart cities, home automation up to healthcare, education and well-being sector. Based on these markets and sectors, IoT can be broadly placed into Consumer IoT (CIoT) and Industrial IoT (IIoT).

CIoT is the technology that we see in our day-to-day lives. It has simplified our lives by automating some of the repetitive daily tasks. Common applications of CIoT are home automation, voice assistant, connected cars, wearable computing, and many more [16]. The IIoT is the industrial variant of IoT used in industry verticals such as manufacturing (Industry 4.0), logistics, transportation, aviation

and so on to enhance the supply chain or optimise production processes. The IIoT distinguishes industrial devices from consumer devices, yet the underlying concept is the same. The goal of IIoT is to reduce human interventions as much as possible in order to maximize automation, and in turn, augment production capacity and efficiency.

Wireless communication technology is playing an important role in both H2H (e.g., Internet) and M2M (e.g., IoT) communication paradigms. There are no one-size-fits-all wireless solutions (or standards) for various industrial as well as consumer applications, as the service requirements and operation environments can vary widely between applications. Different wireless standards described in 1.1.2 can be chosen to best suit the specific use case.

For instance, Zigbee that is based on IEEE 802.15.4 standard can be used for consumer and industrial applications that require short-range and low-rate wireless data transfer. The WirelessHART and ISA100.11a, which are also based on IEEE 802.15.4 and mostly used in IIoT, provide highly reliable, low latency and deterministic data transport, hence suitable for process monitoring and industrial automation [17]. They can support a data rate of up-to 250 kbps. To meet the demands of mission critical applications, the original Carrier Sense Multiple Access (CSMA) procedure in the Media Access Control (MAC) of IEEE 802.15.4 is replaced by a Time Division Multiple Access (TDMA) scheme, which provides deterministic allocations of transmission slots.

Compared with the IEEE 802.15.4 standard, the IEEE 802.11 family addresses the applications demanding higher data rate [18]. The IEEE 802.11 family has received worldwide success in home and office wireless networks. Although enhanced data rates are supported, the early versions of IEEE 802.11 standard (i.e., IEEE 802.11a/g/n/ac) do not provide predictable latency performance, as they rely on a random access scheme like CSMA where for each transmission multiple users have to contend for the shared wireless medium based on a random back-off timer. To reduce latency, IEEE 802.11ax, which is the latest IEEE 802.11 standard, has adopted the Orthogonal Frequency Division Multiple Access (OFDMA) technology. The 802.11ax standard allows more advanced scheduling by implementing OFDMA, which allows multiple users to access the wireless medium at the same time. This reduces the number of contentions and hence provides improved latency performance. The IEEE 802.11ax standard has also introduced more advanced Modulation Coding Scheme (MCS) to further boost the data rate.

In addition to latency and throughput, determinism (i.e., guaranteed packet transport within bounded latency, low packet delay variation, and low packet loss) is crucial for time-critical applications. Table 1.1 shows some use cases that can benefit from determinism in data delivery. For instance, a mobile robot typically relies on communication between the robot and a control system for data transfer (e.g., video, image), guidance control, or eventually emergency stop of the robot

Table 1.1: Use cases for industrial verticals [19, 20].

	Clock Synchronization Accuracy	Bounded Latency
Mobile Robots	$\approx$1 μs	1-10 ms
Closed loop Control	1 μs	1-10 ms
AR/VR	$\approx$1 μs	3-10 ms
Smart Grid	1-20 μs	10-100 ms
Vehicular Communication	10 μs	N.A

in case of danger. In order to achieve timely response of the robot, the control link cannot tolerate more than 10 ms latency. Sometimes multiple robots' movement need to be coordinated in order to fulfill certain tasks, requiring strict time synchronization at sub microsecond level among the robots.

Time Sensitive Networking (TSN) is a set of IEEE 802 standards responsible for providing deterministic connectivity over wired networks. TSN contains four main functions: synchronizing all the clocks on the network using the IEEE 802.1AS standard, prioritizing traffic using the IEEE 802.1Qbv standard, providing reliable links using the IEEE 802.1CB standard, and performing traffic configuration and management tasks using the IEEE 802.1Qcc standard. The foundation of the TSN is based on Clock Synchronization (CS). That is, all the nodes in a distributed network share the same notion of time. Though TSN was originally confined to wired networks, the interests in Wireless TSN (WTSN) is growing, mainly motivated by the recent advances in the standards (i.e., IEEE 802.11 and 5$^{\text{th}}$ Generation (5G) [19]) and the need for wireless connection in time-critical applications. In fact, IEEE 802.1AS in its 2020 release supports network interfaces compliant with the IEEE 802.11 standard. IEEE 802.1AS is a subset (or profile) of Precision Time Protocol (PTP), which is a widely adopted CS protocol in a wired (Ethernet) network. The IEEE 802.1AS-2020 supports media specific interfaces for IEEE 802.3 (Ethernet) and IEEE 802.11 (Wi-Fi). Despite the recent advances in standardization, the requirement of TSN is still rarely met by today's wireless technologies.

The recent mobile operator network 5G has made an attempt to address all the Quality of Service (QoS) requirements, regarding data rate, latency, and determinism. 5G introduces three categories [12]: enhanced Mobile Broadband (eMBB) targets the applications demanding high data rate and large volumes of data transfer with relatively relaxed latency. Examples include video monitoring, video calling, and mobile cloud computing. The massive Machine Type Communication (mMTC) focuses on providing connectivity to a massive number of low-cost and low-power devices with relaxed latency and data rate demands. Common

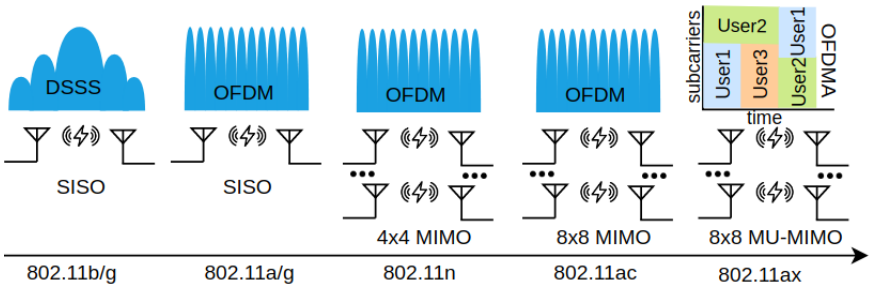


Figure 1.3: Evolution of IEEE 802.11 standard.

applications of mMTC are smart meters, smart homes, wearables, etc. The use of critical Machine Type Communication (cMTC) or Ultra-Reliable Low Latency Communications (URLLC) is for the applications demanding ultra low latency, high reliability, and high availability. Some examples are industrial automation, remote surgery, and public safety. However, the "one-size-fits-all" comes at the price of complexity. Today's 5G network is yet in its early stage: deployments are dominated to eMBB, other categories are still in draft status, and many features specified in the standard are not yet available or will never be available in commercial markets. The reliance on an external mobile operator limits its adoption in private industrial environments. Therefore this PhD research is not directly linked with 5G, however the underlying innovations (such as the reduced hardware footprint) could bring benefits to all modern wireless solutions.

In summary, to address the diverse requirements of numerous applications ranging from consumer oriented applications such as home automation to professional industrial applications such as mobile robots, a plethora of wireless communication standards (or proprietary solutions) have been developed across the world. Further, the wireless communication standards are constantly evolving to address the emerging applications such as the use cases mentioned in Table 1.1.

1.1.4 Evolution of Wireless Communication Standards

The wireless communication standards generally specify the Physical (PHY) and MAC layers of a communication stack, as the remaining layers are in common with the wired network stack. The evolution of the wireless communication standards introduce the technologies and advanced features in order to boost the network performance. For instance, as shown in Fig. 1.3, the IEEE 802.11 legacy standard introduced in 1997 is based on single carrier communication and uses Direct Sequence Spread Spectrum (DSSS) or Frequency Hopping Spread Spectrum (FHSS) to mitigate Inter Symbol Interference (ISI). It is followed by IEEE a/g/n in which higher performance in terms of data rate and latency is achieved, primarily by in-

troducing Orthogonal Frequency Division Multiplexing (OFDM), Multiple Input Multiple Output (MIMO), increasing channel bandwidth and enhanced modulation and coding rates [7]. OFDM is a multicarrier modulation technique used to mitigate ISI caused by multipath propagation and narrowband interference, two channel impairments that readily occur in a shared Industrial, Scientific and Medical (ISM) spectrum band and indoor environments. MIMO is an antenna technology used to enhance either the throughput through spatial multiplexing or the reliability through spatial diversity (e.g., beamforming). All of these standards however can only service a single User (SU) at a time, meaning that the wireless medium can only be occupied by a Station (STA) either transmitting to or receiving from an Access Point (AP). IEEE 802.11ac introduced the Down Link (DL) Multi User (MU) MIMO, i.e., an AP is now able to transmit to multiple STAs simultaneously. The IEEE 802.11ax standard [8] took a step forward and introduced both DL and Up Link (UL) MU transmission. The key enablers for the MU transmission are OFDMA and MU-MIMO. OFDMA is the MU version of OFDM which allows simultaneous MU transmission in the same frequency channel by assigning subsets of Subcarriers (SCs) named Resource Unit (RU) to individual users. Though the given details are specific to IEEE 802.11 family, OFDMA and MU-MIMO are also applied for other standards such as IEEE 802.16 (WiMAX), LTE, 5G NR [9, 12]. All these standards are build on top of OFDM, so far the most widely used wireless technology with performance close to the theoretical Shannon limit, and leverage on similar techniques to boost performance in their further evolutions.

On one hand, realization of these advanced features of the standards require complex Hardware (HW) designs for for Baseband Processing Units (BBPUs) of the respective radio Transceiver (TRX). On the other hand, realization of the wireless standards also demand flexible, reconfigurable and programmable platforms, as new advanced features are constantly being introduced. Thus, HW efficient designs for the radio TRXs on flexible, reconfigurable and programmable platforms are higly desirable for modern wireless communication systems. An SDR is an attractive platform for designing the radio TRXs of the modern wireless communication systems, as a state-of-the art SDR platform is not only flexible, but is equipped with high processing capabilities.

1.2 Software Defined Radio

In 90's, Joseph Mitola coined the SDR term and envisioned an ideal SDR, whose physical components were only an antenna, an Analog-to-Digital Converter (ADC) and a Digital-to-Analog Converter (DAC). Rest of the radio functions would be handled by a reprogrammable processor [22]. Just like wireless communication systems, a SDR has also evolved from military use to commercial applications.

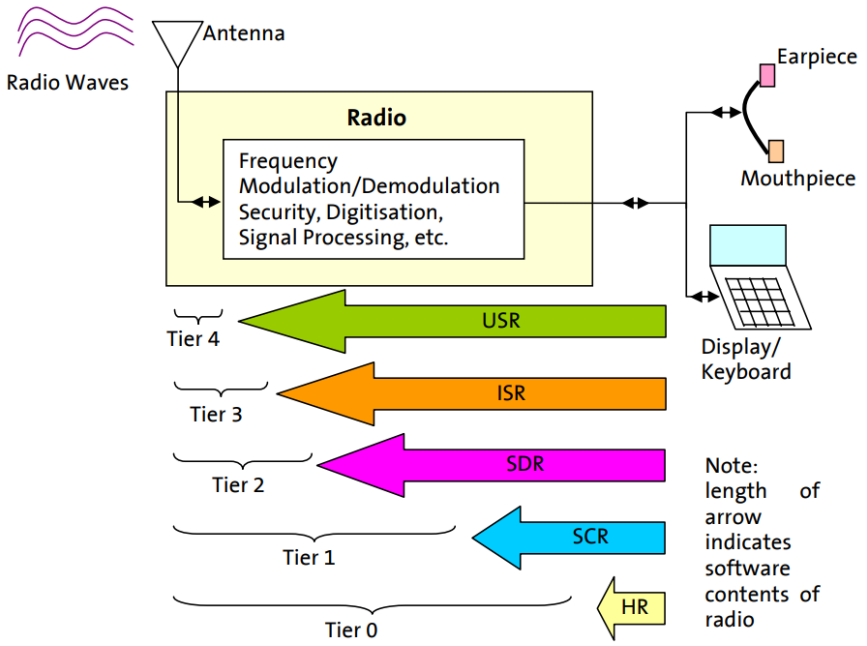


Figure 1.4: High-level Abstraction of the SDR tier definition, where the length of arrow represents the amount of software defined functions in a radio [21].

The Navy of the United States introduced the first operational SDR known as Speakeasy in 1995 [23]. The concept of a SDR has evolved over the past three decades and a number of SDR's definitions are proposed during this period.

1.2.1 Definitions of Software Defined Radio

The SDR Forum, working collaboratively with the IEEE P1900.1 group, has developed a multi-tiered definition of an SDR. The five tier definition, which provides a clear overview of the term and its associated benefits [21], of an SDR is illustrated in Fig. 1.4 and summarized as follows:

- Tier 0: A non-configurable Hardware Radio (HR), which acts as a baseline radio, whose functionality can not be changed by software.
- Tier 1: A Software Controlled Radio (SCR) in which some or all the radio functions are controlled by software. The radio chain of the SCR is implemented using Application Specific Integrated Chip (ASIC). In other words, the implementation of radio functions is fixed and a software interface may allow to change certain parameters of radio functions such as frequency and transmit power.

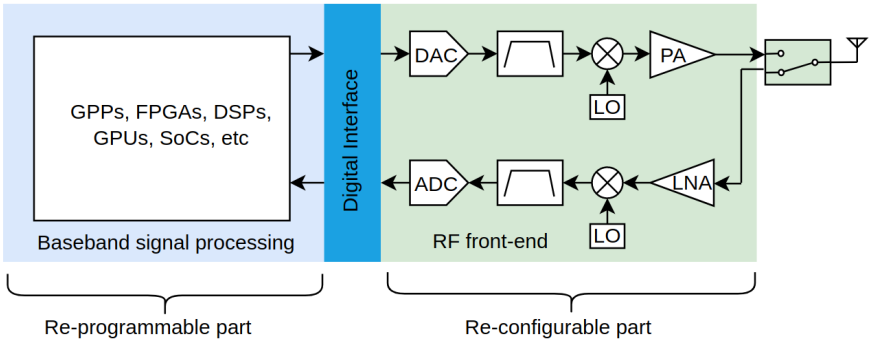


Figure 1.5: Block diagram of an SDR architecture.

- Tier 2: An SDR in which some or all the radio functions are defined by software. The SDR still has radio functions which are software controllable including frequency, transmit power level and narrow/wide band operation. The Radio Frequency (RF) front-end is HW based and non-configurable in the SDR.

- Tier 3: An Ideal Software Radio (ISR) implements much more radio functions in the digital domain with only an ADC, a DAC and antenna as HW components. An ISR is fully programmable and reconfigurable.

- Tier 4: An Ultimate Software Radio (USR) goes beyond the ISR. USR does not only have fully programmable and reconfigurable radio, but also supports the broad range of radio functions and frequencies at the same time. The USR can switch from one air-interface/application to another in milliseconds.

The exact definition of SDR is ambiguous, the tiered definition provides a way to visualize different levels of software control an SDR may possess. The features of today's commercial SDRs are most similar to the Tier 2 SDR. A commercial SDR is non-technology specific, and is composed of two parts: a reprogrammable part that usually consists of generic signal processing unit(s) for realizing digital baseband processing functions, and a reconfigurable part that contains the analog RF front-end, as shown in Fig. 1.5. The generic signal processing units may include Field Programmable Gate Arrays (FPGA), Digital Signal Processor (DSP), Graphics Processing Unit (GPU), General Purpose Processor (GPP), programmable System on Chip (SoC) or other application specific programmable processors. The RF front-end normally consists of upconverters and power amplifiers on the Transmitter (TX) side, and Low Noise Amplifiers (LNAs) and downconverters on the Receiver (RX) side. The ADC and DAC, operating at baseband frequencies or

intermediate frequencies, act as a bridge between analog and digital domains. Although the the RF front-end is in the analog domain, it still provides the software configurable features such as transmit power, carrier frequency, channel Bandwidth (BW).

1.2.2 Types of Software Defined Radio

SDRs can be broadly classified into two groups: GPP based SDR, and non-GPP based SDR.

1.2.2.1 GPP based SDR

The GPP based SDR is easy to adopt, as it allows to realize the radio functions entirely using software. In GPP based SDR, a Personal Computer (PC) generally receives/transmits the digitized baseband samples to/from RF front-end over Ethernet, Universal Serial Bus (USB), or Peripheral Component Interconnect (PCI) interfaces. The inherent flexibility and availability of GPPs make them the best candidates for SDR platforms in a research lab environment, where size, weight and power consumption are not the major concerns. GPP based SDRs are suitable for communication systems demanding relatively low throughput (or data rate) in real time. Examples of GPP based SDRs are IMEC's BEAR Platform, Sandberg SB3500, Infeneon Music, and Universal Software Radio Peripheral (USRP) [24]. One of the major concerns of GPP based SDRs is the latency and jitter introduced by the Operating System (OS), which generally exceeds 10 ms on Windows system, and under 1 ms on a real-time OS [25]. Furthermore, the interface between GPP and RF front-end introduces additional latency and jitter, which is under 100 μs for Ethernet [26] and 200 μs for USB [27]. Such type of SDRs are hence no suitable candidates for wireless communication systems requiring low latency and low jitter processing in real time. GPP based SDRs are also no feasible candidates for wireless communication systems, which require the transmission to be scheduled within a bounded time based on the received data. Many wireless communication standards place tight requirements on the response time upon packet reception. For instance, Long Term Evolution (LTE) [12] needs to schedule an Acknowledgment (ACK) packet within 4 ms (when operating in Frequency Division Duplex (FDD) mode) after the successful reception of a data packet.

In order to achieve high data rate processing in real time, the GPP based SDR can offload the radio functions (e.g., (de)modulation, (de)encoding, etc) to Hardware Accelerator (HA). An HA is used when some of the radio functions can not be executed in real time on GPP based SDR. For instance, these radio functions can be offloaded to an FPGA-based HA. An FPGA is an integrated circuit designed to be configured by a designer after manufacture. Xilinx (or Advanced Micro Devices (AMD)) and Altera (or Intel) are the major manufactures of FPGA

fabrics. USRP [28], which is the most widely used SDR platform in the research community, is used in tandem with a host PC, where the GPP is running on the host PC, and the FPGA-based HA and RF front-end are included in the USRP platform. Depending on the exact type of USRP, it may interact with the host PC via PCIe, Ethernet, or USB 3.0 interfaces. Such hybrid SDR, in which two signal processing units are on two different chips (i.e., GPP is running on host PC, and FPGA on USRP), provides improved data rate, but still suffers from latency and jitter introduced by the interfaces (i.e., 79 μs for PCIe, 66 μs for USB 3.0, and 106 μs for 10 Gbps Ethernet [29]).

1.2.2.2 non-GPP based SDR

The non-GPP based SDR is a type of SDR in which all signal processing units are embedded on the SDR platform itself. Comparing with GPP based SDR, the interfaces between processing units on a non-GPP based SDR are faster, resulting in smaller latency and jitter, hence resulting in better real-time performance.

The Xilinx's Zynq SoC together with Analog Devices' AD9361 RF front-end forms a family of popular SDR platforms, which is the state-of-the-art for non-GPP based SDRs. The Zynq SoC combines the Advanced RISC Machines (ARM) processor and FPGA fabric on the same chip. Zynq SoC uses Advanced eXtensible Interface (AXI) high speed buses between ARM processor and FPGA. AXI interface offers low latency (i.e., 1.435 μs [29]) and high throughput performance. Another example of non-GPP based SDRs is the USRP Embedded series (or USRP E-Series). Though both the FPGA and processor on the USRP E-series are less powerful than the Zynq SoC.

This PhD book focuses on providing HW efficient designs for low latency, deterministic wireless communication system using SDR. The Zynq SoC based SDR is chosen during this PhD work, as such kind of SDR satisfies the requirements for low latency interface (i.e., AXI) between processing units, high throughput HA on FPGA[1], and configurable RF front-end.

1.3 Motivation and Challenges

Since the birth of the first wireless communication system, wireless technology has been constantly evolving to address the ever increasing and vastly diverging use cases. Section 1.1 described the evolution of wireless communication systems and the complexity involved in designing the corresponding radio TRXs. Section 1.2 introduced the concept of SDR, which allows different levels of control on the

[1]Zynq Ultrascale+ ZCU102 can provide a maximum Digital Signal Processing performance of 5,491 Giga Multiply-Accumulates per Second (GMACS) [30].

radio TRX by the users. In addition to the flexibility, modern SDR devices also should possess high performance in terms of processing capacity and latency.

As elaborated in Sections 1.1 & 1.2, it is obvious that adding new functionalities such as MU communication in wireless communication systems increases the complexity and thereby the HW footprint of the corresponding radio TRXs. Hence, more HW efficient designs for the modern radio TRX are highly desired. Furthermore, there is also a need for realizing a radio TRX on a flexible HW platform such as a SDR, in order to keep up with the ever evolving wireless standards.

Therefore, this PhD dissertation first provides hardware efficient designs for MU radio TRXs, and then presents concrete techniques to optimize TRXs' performance when implemented on a SDR platform. Finally, this dissertation presents an cost-effective solution for high-accuracy CS over a wireless network, which is a fundamental feature required by many time-critical applications, but not yet available in today's commercial off-the-shelf wireless systems. The remainder of this section presents more details on the motivation and challenges for each of the contributions.

1.3.1 Reduction of Hardware Footprint for Multi-User Wireless Communication Systems

In an IoT (or IIoT) wireless communication network, a Gateway (GW) is mostly equipped with a SU TRX and therefore is only able to communicate with a single end-device (or STA) at a time. Since the number of end-devices connected with the GW in IoT network has proliferated significantly, A SU TRX based GW adversely affects the performance specifically in terms of latency of an IoT network when multiple smart devices try to access the GW at the same time. One way to improve the performance of the GW is to use a MU TRX. Many wireless communication standards such as IEEE 802.11ax support MU communication by introducing MU-OFDMA and MU-MIMO technologies. However, the MU functionality poses new challenges in the wireless chip design. Existing MU TRXs rely on the duplication approach wherein a dedicated HW processing block is utilized per user. The HW footprint of such an approach increases proportionally with the number of users simultaneously served by the MU TRX. Therefore, there is a need of an alternative HW efficient design that should ideally use the same HW footprint of a single user irrespective to the number of users being served by a MU TRX, and should hold the same performance as the duplication approach does. This is a challenging but urgent task, given the increased popularity of MU TRX in recent standards.

1.3.2 Reduction of Turnaround Time of the RF Front-end on Modern SDR

The wireless communication standards such as IEEE 802.11 and 5G operating in Time Division Duplex (TDD) have strict latency requirements. For instance, IEEE 802.11 standard needs to send an ACK packet within 10 microseconds (when operating in 2.4 GHz frequency band) after the successful reception of a data packet. The self-contained TDD operation in 5G New Radio (NR) also need to return an ACK packet within a few microseconds. Apart from meeting stringent timing requirements of MAC protocols such as fast acknowledgements, a low Turnaround Time (TT) is also critical to reduce the probability of collisions in Listen Before Talk (LBT) like MAC operation. An SDR is the most promising platform for designing the radio TRXs of modern wireless communication systems, as state-of-the-art SDR platforms are not only flexible, but also support high digital processing capabilities. However, the TT of the analog RF front-ends of such SDR platforms easily takes tens of microseconds, because during switching between transmit and receive modes, powering up and stabilizing of components (e.g., amplifier, mixer, ADC and DAC) in the analog RF front-end consume a considerable amount of time. Modern SDRs can be used to design high-performance wireless communication standards operating in FDD mode (e.g., WiMAX, LTE, and 5G) or to design wireless communication standards operating in TDD mode with relaxed requirement of TT (e.g., IEEE 802.15.4). However, the slow TT of the analog RF front-ends constrain the design of novel high-performance wireless communication standards operating in TDD mode (e.g., 5G, IEEE 802.11). There is clearly need to further reduce the TT of the analog RF front-end to achieve switching times as close as possible to zero.

1.3.3 Achieving High Precision Clock Synchronization over Wireless Network

Low latency is just one aspect for achieving deterministic wireless connectivity serving time-critical industrial applications such as mobile robots or close-loop control. A TDMA MAC scheme as used in some wireless communication standards has the potential to provide deterministic allocation of transmission slots under the condition that all devices participating in the network are perfectly synchronised in time. Accurate CS is hence the foundation for realizing deterministic TDMA schedules. The de facto standard for deterministic connectivity is TSN. However, the original specification only targets wired networks (more specifically Ethernet). Recent advances in WLAN and 5G standards have generated significant interest in extending the wired TSN capabilities to the wireless domain. The TSN standard is based on the principle that all the nodes in a distributed network share the same notion of time, which is possible using CS protocols. In fact, the CS

standard (i.e., IEEE 802.1AS) of TSN in its 2020 release also supports the IEEE 802.11 standard interface. IEEE 802.1AS is a profile of PTP. Until today, most of the PTP based networks have been implemented over the wired network [31–33]. The CS accuracy of PTP depends on the timestamps captured at the moment of packet reception or transmission. TimeStamping (TS) in PTP can be achieved via HW or Software (SW). HW TS is produced at or close to the physical layer, it is realized by using dedicated HW to assist the TS. On the other hand, SW TS is generated in device drivers or at a higher layer of network stack without HW assistance. The CS accuracy is significantly affected by the TS location and it is recommended to place the TS location as close as possible to the physical layer, so that the TS is least affected by the time variation of packets when travelling through the network stack. PTP has recently gained lots of attention in the wireless community as the way to extend TSN to the wireless domain (further referred to as WTSN). However, existing PTP designs for wireless networks reported in literature either incorporate SW TS delivering poor CS accuracy, or HW TS providing better synchronization accuracy at the cost of a significant HW overhead. So far, there is no commercial wireless solution that provides accurate CS due to lack of HW TS support. There is hence a clear need for a CS solution, which provides better synchronization accuracy without using a dedicated clock (or extra HW) for HW TS, which is a challenging task.

1.4 Outline and Research Contributions

This dissertation is composed of a number of research publications performed during the period of this PhD. The publications cover the integral part of the research work. Only minor changes are done to the original manuscripts of the published papers to address grammatical and formatting issues. The complete list of publications produced from this research work is listed in Section 1.5. Fig. 1.6 positions the different contributions that are presented in each chapter (Ch.). In this section, the outline of the remainder of this dissertation is detailed by highlighting the research contribution of each chapter.

Chapter 2 begins by mentioning the adverse impact of a GW equipped with a SU TRX on an IoT (or IIoT) network's performance (particularly in terms of latency and throughput). Next, the advantage of replacing a SU TRX based GW with a MU TRX based GW in a highly congested IoT network is detailed. Then, the hardware duplication and hardware virtualization approaches to design a MU TRX are analysed. Finally, **a novel hardware virtualization based design for a MU TRX is introduced. The virtual TRX offers MU capabilities and uses the same the physical HW resources of a SU TRX with some extra HW overhead to manage the virtualization process, which is negligible compared to the resources consumed by a SU TRX.**

MU communication can be broadly categorized into two types. In the first type of MU communication, the capacity of the system increases with the number of users simultaneously participating in the MU communication. This is the case for concurrent multi-channel communication and MU-MIMO. When designing such a MU TRX using hardware virtualization, the operating clock frequency of the final design will increase proportionally with the number of users served in the MU TRX. The proposed hardware design for MU TRX in Chapter 2 is an example of such a MU communication system. In the second type of MU communication, the capacity of the system remains fixed irrespective to the number of users, or in other words, the capacity is shared among multiple users. For instance, MU-OFDMA allows multiple users to simultaneously access the same channel by allocating subsets of SCs to different users. **Chapter 3 introduces a novel HW efficient design for an OFDMA based MU TRX, which has comparable HW footprint of a SU TRX regardless of the number of users being served, thanks to the hardware virtualization technique.** The applicability of the design is validated for an IEEE 802.11ax compliant OFDMA transmitter on an FPGA of a modern SDR platform. The performance and HW footprint is compared against the conventional duplication approach.

Chapter 4 starts with describing the importance of SoC based SDR for designing and experimentally validating state-of-the-art wireless communication standards. Then, the limitations of modern SDR platforms when used to validate high-performance radios operating in TDD mode are highlighted by unveiling the slower TT of their analog RF front-ends when compared against their ASIC counterparts. Finally, **a novel solution, in which the BBPU operates in half-duplex mode and the analog RF front-end operates in full-duplex mode, is presented to reduce the TT on a SDR. The proposed solution is experimentally validated on the widely adopted AD9361 RF front-end.**

Chapter 5 targets the HW efficient design of a CS algorithm, which is a critical component for a deterministic network (e.g., when applying a TDMA based scheduling mechanism in a TSN). First, an overview is given of the workflow of the PTP protocol in a distributed network. Then, SW TS and HW TS based PTP designs for wireless networks are analysed. Finally, **a novel SDR based approach to implement PTP using HW TS is presented. Instead of adding a dedicated HW clock for synchronization purposes, the solution makes use of the existing radio clock as specified in the communication standard.** The proposed solution is experimentally validated for IEEE 802.11 WLAN, which utilizes the Timing Synchronization Function (TSF) clock, an existing clock in IEEE 802.11 standard for synchronization between an access point and WLAN stations. Microsecond level synchronization accuracy has been demonstrated within a single-hop WLAN as well as across wired-wireless networks.

For simplifying navigation through this PhD dissertation, Table 1.2 maps the

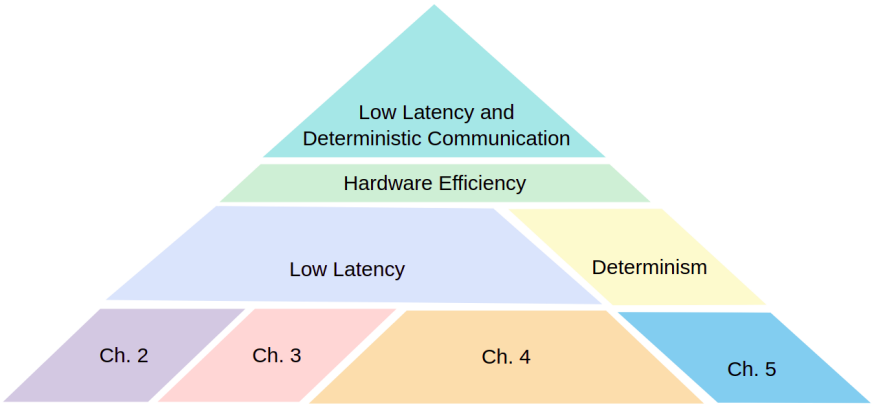


Figure 1.6: Schematic position of the different chapters in this dissertation

challenges that are highlighted in Section 1.3 to the chapters where the challenges are addressed.

Although the main focus of this PhD dissertation is to offer HW efficient radio designs for deterministic and low latency wireless communication system using SDR, the flexible feature of an SDR also enables modern wireless communication standards (particularly standards having fixed bandwidth such as IEEE 802.15.4) to adopt optimal performance in a dynamic radio spectrum environment (as is the case in the 2.4 GHz ISM band that is shared by many technologies) by changing their bandwidth at run-time (e.g., narrow-band modes fit best in a crowded spectrum, whereas wide-band modes perform better for low latency scenario in less dense scenarios). For the sake of completeness, the Appendix A provides an SDR based IEEE 802.15.4 compliant flexible HW design in which the bandwidth and consequentially the data rate and latency can be configured at run-time according to the spectral conditions. Furthermore, in the work of Chapter 3 and Chapter 5, $openwifi$—an existing open-source Wi-Fi implementation on modern SDR platforms—is made use and modified. Therefore, Appendix B describes the position of the work done in the Chapter 3 and Chapter 5 with respect to $openwifi$.

Table 1.2: An overview of the contributions per chapter in this dissertation.

	Ch.2	Ch.3	Ch.4	Ch.5
Reduction of Hardware Footprint for Multi-User Wireless Communication Systems	•	•		
Reduction of Turnaround Time of the RF Front-end on Modern SDR			•	
Achieving High Precision Clock Synchronization over Wireless Network				•

1.5 Publications

The research results obtained during this PhD research have been published in scientific journals and presented at a series of international conferences. The following list provides an overview of the publications during this PhD research.

1.5.1 Publications in international journals (listed in the Science Citation Index [2])

1. **Muhammad Aslam**, Xianjun Jiao, Wei Liu, Ingrid Moerman. *An approach to achieve zero turnaround time in TDD operation on SDR front-end.* Published in IEEE Access, November 2018. **IF: 3.36**

2. **Muhammad Aslam**, Xianjun Jiao, Wei Liu, Ingrid Moerman. *CMCVT: A Concurrent Multi-Channel Virtual Transceiver.* Published in AEU - International Journal of Electronics and Communications, June 2020. **IF: 3.18**

3. **Muhammad Aslam**, Wei Liu, Xianjun Jiao, Jetmir Haxhibeqiri, Gilson Miranda, J. Hoebeke, Johann M. Marquez-Barja, Ingrid Moerman. *Hardware Efficient Clock Synchronization Across Wi-Fi and Ethernet-Based Network Using PTP.* Published in IEEE Transactions on Industrial Informatics, June 2022. **IF: 10.2**

1.5.2 Publications in international conferences (listed in the Science Citation Index [3])

1. Jetmir Haxhibeqiri, Xianjun Jiao, **Muhammad Aslam**, Ingrid Moerman, Jeroen Hoebeke. *Enabling TSN over IEEE 802.11: Low-overhead Time Synchronization for Wi-Fi Clients*. Published in 22nd IEEE International Conference on Industrial Technology, 10-12 March 2021. p. 1068-1073. Valencia, Spain.

2. **Muhammad Aslam**, Xianjun Jiao, Wei Liu, Michael Mehari, Thijs Havinga, Ingrid Moerman. *A Novel Hardware Efficient Design for IEEE 802.11ax compliant OFDMA Transceiver using Hardware Virtualization.* Submitted

[2] The publications listed are recognized as 'A1 publications', according to the following definition used by Ghent University: A1 publications are articles listed in the Science Citation Index Expanded, the Social Science Citation Index or the Arts and Humanities Citation Index of the ISI Web of Science, restricted to contributions listed as article, review, letter, note or proceedings paper.

[3] The publications listed are recognized as 'P1 publications', according to the following definition used by Ghent University: P1 publications are proceedings listed in the Conference Proceedings Citation Index - Science or Conference Proceedings Citation Index - Social Science and Humanities of the ISI Web of Science, restricted to contributions listed as article, review, letter, note or proceedings paper, except for publications that are classified as A1.

in the 28th Annual International Conference on Mobile Computing and Networking (MOBICOM 2022), 17-21 October 2022, Sydney, Australia.

1.5.3 Publications in Other International Conferences

1. **Muhammad Aslam**, Wei Liu, Xianjun Jiao, Jetmir Haxhibeqiri, Jeroen Hoebeke, Ingrid Moerman . *High precision time synchronization on Wi-Fi based multi-hop network.* Published in 2021 IEEE INFOCOM 2021 - IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS), 10-13 May 2021. p. 1-2. Vancouver, BC, Canada.

2. Xianjun Jiao, **Muhammad Aslam**, Wei Liu, Ingrid Moerman. *An antenna switching based NOMA scheme for IEEE 802.15.4 concurrent transmission.* Published in 2020 IEEE 91st Vehicular Technology Conference (VTC2020-Spring), 25-28 May 2020. p. 1-5. Antwerp, Belgium.

3. Xianjun Jiao, Wei Liu, Michael Mehari, **Muhammad Aslam**, Ingrid Moerman. *Openwifi : a free and open-source IEEE802.11 SDR implementation on SoC*. Published in 2020 IEEE 91st Vehicular Technology Conference (VTC2020-Spring), 25-28 May 2020. p. 1-2. Antwerp, Belgium.

4. **Muhammad Aslam**, Xianjun Jiao, Wei Liu, Ingrid Moerman . *An enhanced version of IEEE 802.15.4 standard compliant transceiver supporting variable data rate.* Published in (2019) EuCNC 2019 : Enabling smart connectivity : proceedings., 18-21 June 2019. p. 1-2. Valencia, Spain.

1.5.4 Patent Applications

1. **Muhammad Aslam**, Xianjun Jiao, Wei Liu, Ingrid Moerman . *Device and method for clock synchronisation in a wireless network.* Filed in the European Patent Office (EPO) on 8 October 2021. European Patent Application EP 21201783.4.

References

[1] G. Andrea. *Wireless communication*. 2017.

[2] C. A. Tavera, J. H. Ortiz, O. I. Khalaf, D. F. Saavedra, and T. H. Aldhyani. *Wearable wireless body area networks for medical applications*. Computational and Mathematical Methods in Medicine, 2021, 2021. doi:https://doi.org/10.1155/2021/5574376.

[3] *IEEE Standard for Low-Rate Wireless Networks*, 2020. Available from: https://standards.ieee.org/ieee/802.15.4/7029/.

[4] *IEEE Standard for Information technology– Local and metropolitan area networks– Specific requirements– Part 15.1a: Wireless Medium Access Control (MAC) and Physical Layer (PHY) specifications for Wireless Personal Area Networks (WPAN)*, 2005. Available from: https://standards.ieee.org/ieee/802.15.1/3513/.

[5] *IEEE Standard for High Data Rate Wireless Multi-Media Networks*, 2016. Available from: https://standards.ieee.org/ieee/802.15.3/6211/.

[6] *IEEE Standard for Information technology–Telecommunications and information exchange between systems - Local and metropolitan area networks–Specific requirements - Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications Amendment 2: Sub 1 GHz License Exempt Operation*, 2016. Available from: https://standards.ieee.org/ieee/802.11ah/4960/.

[7] *802.11-2020 - IEEE Standard for Information Technology–Telecommunications and Information Exchange between Systems - Local and Metropolitan Area Networks-Specific Requirements - Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications.*, 2020.

[8] *802.11ax-2021 - IEEE Standard for Information Technology–Telecommunications and Information Exchange between Systems Local and Metropolitan Area Networks–Specific Requirements Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications Amendment 1: Enhancements for High-Efficiency WLAN*, 2021.

[9] *802.16-2017 - IEEE Standard for Air Interface for Broadband Wireless Access Systems*, 2017. Available from: https://ieeexplore.ieee.org/document/8303870.

[10] *Lora Alliances*, 2022. Available from: https://www.lora-alliance.org/what-is-lora/technology.

[11] *Sigfox*, 2022. Available from: https://www.sigfox.com/=.

[12] *The Mobile Broadband Standard*, 2022. Available from: https://www.3gpp.org/.

[13] *Mobile Satellite Communication Standard*, 2022. Available from: https://www.etsi.org/.

[14] L. Tan and N. Wang. *Future internet: The Internet of Things*. In 2010 3rd International Conference on Advanced Computer Theory and Engineering(ICACTE), volume 5, pages 376–380, 2010. doi:10.1109/ICACTE.2010.5579543.

[15] *IoT connections outlook*, 2021. Available from: https://www.ericsson.com/en/reports-and-papers/mobility-report/dataforecasts/iot-connections-outlook.

[16] T. Alladi, V. Chamola, B. Sikdar, and K.-K. R. Choo. *Consumer IoT: Security vulnerability case studies and solutions*. IEEE Consumer Electronics Magazine, 9(2):17–25, 2020. doi:https://doi.org/10.1109/MCE.2019.2953740.

[17] *IEEE Standard for High Data Rate Wireless Multi-Media Networks*, 2020. Available from: https://www.emerson.com/documents/automation/white-paper-industrial-iot-devices-wirelesshart-5g-en-6872874.pdf.

[18] V. K. Huang, Z. Pang, C.-J. A. Chen, and K. F. Tsang. *New Trends in the Practical Deployment of Industrial Wireless: From Noncritical to Critical Use Cases*. IEEE Industrial Electronics Magazine, 12(2):50–58, 2018. doi:10.1109/MIE.2018.2825480.

[19] *Avnu Wireless TSN White Paper: Market Expectations, Capabilities & Certification*, 2022. Available from: https://avnu.org/wireless-tsn-white-paper-2-download/.

[20] I. Val, O. Seijo, R. Torrego, and A. Astarloa. *IEEE 802.1AS Clock Synchronization Performance Evaluation of an Integrated Wired–Wireless TSN Architecture*. IEEE Transactions on Industrial Informatics, 18(5):2986–2999, 2022. doi:10.1109/TII.2021.3106568.

[21] T. Sturman, A. Burr, J. Fitzpatrick, T. James, M. Rupp, and S. Weiss. *An evaluation of software defined radio-main document*. In OFCOM R&D Symposium, 2006. doi:http://dx.doi.org/10.13140/RG.2.1.4454.2800.

[22] *SDRF Cognitive Radio Definitions*, 2007. Available from: http://www.sdrforum.org/pages/documentLibrary/documents/SDRF-06-R-0011-V1_0_0.pdf.

[23] R. Lackey and D. Upmal. *Speakeasy: the military software radio*. IEEE Communications Magazine, 33(5):56–61, 1995. doi:10.1109/35.392998.

[24] M. Palkovic, P. Raghavan, M. Li, A. Dejonghe, L. Van der Perre, and F. Catthoor. *Future Software-Defined Radio Platforms and Mapping Flows*. IEEE Signal Processing Magazine, 27(2):22–33, 2010. doi:10.1109/MSP.2009.935386.

[25] E. Grayver. *Implementing software defined radio*. Springer Science & Business Media, 2012.

[26] R. Hughes-Jones, P. Clarke, and S. Dallison. *Performance of 1 and 10 Gigabit Ethernet cards with server quality motherboards*. Future Gener. Comput. Syst., 21:469–488, 2005. doi:https://doi.org/10.1016/j.future.2004.10.002.

[27] T. Schmid, O. Sekkat, and M. B. Srivastava. *An experimental study of network performance impact of increased latency in software defined radios*. In Proceedings of the second ACM international workshop on Wireless network testbeds, experimental evaluation and characterization, pages 59–66, 2007. doi:https://doi.org/10.1145/1287767.1287779.

[28] *USRP products*, 2022. Available from: https://www.ettus.com/products/.

[29] X. Jiao, I. Moerman, W. Liu, and F. A. P. de Figueiredo. *Radio Hardware Virtualization for Coping with Dynamic Heterogeneous Wireless Environments*. In P. Marques, A. Radwan, S. Mumtaz, D. Noguet, J. Rodriguez, and M. Gundlach, editors, Cognitive Radio Oriented Wireless Networks, pages 287–297, Cham, 2018. Springer International Publishing. doi:https://doi.org/10.1007/978-3-319-76207-4_24.

[30] *UltraScale Architecture and Product Data Sheet: Overview*, 2021. Available from: https://www.xilinx.com/support/documentation/data_sheets/ds890-ultrascale-overview.pdf.

[31] G. Gaderer, R. Holler, T. Sauter, and H. Muhr. *Extending IEEE 1588 to fault tolerant clock synchronization*. In IEEE International Workshop on Factory Communication Systems, 2004. Proceedings., pages 353–357. IEEE, 2004. doi:https://doi.org/10.1109/WFCS.2004.1377745.

[32] *CIP Sync: an Ethernet based commercial product compliant with IEEE 1588 standard*. Available from: https://www.odva.org/technology-standards/distinct-cip-services/cip-sync/.

[33] J. Feld. *PROFINET-scalable factory communication for all applications*. In IEEE International Workshop on Factory Communication Systems, 2004. Proceedings., pages 33–38. IEEE, 2004. doi:https://doi.org/10.1109/WFCS.2004.1377673.

2

CMCVT: A Concurrent Multi-Channel Virtual Transceiver

As previously discussed, one way to improve the performance in terms of latency of Internet of Things (IoT) or Industrial IoT (IIoT) network is to use Gateway (GW) equipped with Multi User (MU) Transceiver (TRX). However, the MU functionality poses new challenges in the chip design. Existing MU TRXs rely on the duplication approach wherein a dedicated Hardware (HW) is utilized per user. The HW footprint of such an approach increases proportionally with the number of users simultaneously served in the MU TRX. In this chapter, a novel HW virtualization based design for MU TRX is introduced and demonstrated for an IEEE 802.15.4 based 8×8 channel TRX. The novel design for MU TRX has comparable HW footprint of a single User (SU) TRX regardless of the number of users being served.

⋆ ⋆ ⋆

Muhammad Aslam, Xianjun Jiao, Wei Liu, and Ingrid Moerman.

Published in AEU - International Journal of Electronics and Communications, Vol. 6, pp. 1-8, June 2020.

Abstract State-of-the-art wireless Gateways (GWs) used in Internet of Things (IoT) offer a single channel radio link, which limits the capabilities of the IoT

network controlled by the GW, as the GW can only use a single channel at a time to communicate with the end-device(s). The quality of service (e.g., aggregate throughput, latency) offered by a single channel GW could be substantially improved by employing a multi-channel transceiver, which is capable of transmitting/receiving data on different radio channels simultaneously, particularly for larger wireless networks. However, current solutions available in both research and commercial communities only offer multi-channel receiver capabilities, and do not incorporate the multi-channel transmitter part. In addition, in terms of implementation, these multi-channel receivers duplicate single-channel hardware functionality. In this chapter, for the first time, a novel concurrent multi-channel virtual transceiver is introduced. The virtual transceiver offers multi-channel capabilities and uses the same single-hardware hardware implementation for the Physical (PHY) layer by employing the virtualization technique. This new virtual transceiver concept is demonstrated for an IEEE 802.15.4 based 8×8 channel transceiver, implemented on an Field Programmable Gate Arrays (FPGA) of a modern Software Defined Radio (SDR) and is compared with the existing duplication approach. The duplication approach consumes 9008 Look Up Tables (LUTs), and 12120 Flip-Flops (FFs), whereas the proposed approach occupies only 2959 LUTs and 2105 FFs, saving 67.15% LUTs and 82.63% FFs in comparison with the duplication approach. The experimental results reveal that the virtual transceiver provides the same performance (e.g., receiver sensitivity of -98.5 dBm) as the transceiver achieved by duplicating the PHY layers but consumes much less hardware resources.

2.1 Introduction

Internet of Things (IoT) is a technology that interconnects things (e.g., objects, machines, etc.) and allows them to exchange information with each other with or without human intervention. No matter what type of IoT applications are considered, the IoT infrastructure in general consists of: (1) smart end-devices capable of processing, sensing, and actuating the environment, which are connected to the cloud via a network device with gateway capabilities (further referred to as IoT Gateway (GW)), (2) a IoT GW which manages the bidirectional traffic between smart devices and the cloud (or internet), (3) an data server infrastructure in the cloud responsible for storing, analysing, and processing the huge amount of data in real time, and (4) a user interface which is tangible, visible and accessible by users [1].

While the number of end-devices connected with the Internet in IoT network has proliferated significantly, currently, an IoT GW is the sole way of connecting them with the internet. It is mostly equipped with a single channel Transceiver (TRX), which is only capable of communicating with end-device(s) over a single

channel at a time, which often appears to be a bottleneck. The single-channel TRX is adequate for point-to-point communication or for the scenario in which the number of smart devices directly connected with the IoT GW is limited. A single channel TRX based GW adversely affects the performance (in particular in terms of latency and throughput) of an IoT network when multiple smart devices try to access the GW at the same time. One way to improve the performance of the GW is to use concurrent multi-channel TRX. The advantage of the multi-channel TRX is that it has the capacity to transmit/receive data on more than one channel simultaneously. Such a multi-channel TRX will certainly outperform a single-channel TRX when a large number of devices are connected. It is important to note that the multi-channel TRX is different from a multi-band TRX. A concurrent multi-band TRX operates simultaneously in different frequency bands (e.g., 2.4 GHz or 5 GHz band), but utilizes a single channel in each band at a time [2]. While, the multi-channel TRX utilizes multiple frequency channels within a single frequency band.

There are already many multi-channel radio sniffers available in the IoT networks, which can receive data on multiple channels simultaneously. These sniffers are achieved either by using many single channel commercial off-the-shelf chips or by duplicating the same HW on a Software Defined Radio (SDR) [3, 4]. An SDR is a radio communication system on which the radio's components are implemented in software on a host-computer or on a programmable hardware device (e.g., Field Programmable Gate Arrays (FPGA), etc.). Since the sole purpose of these multi-channel solutions is sniffing packets on the channels, they do not incorporate a multi-channel transmitter. The solutions do not only lack multi-channel transmitter part, but also underutilize the potential processing capabilities of a modern SDR or Application Specific Integrated Chip (ASIC). For example, the parallel processing feature of the FPGA in a modern SDR enables it to process the data at a rate far higher (e.g., the FPGA in Zynq 7000 can provide a maximum DSP performance of 3,634 Giga Multiply-Accumulates per Second (GMACS) [5]) than that required by the wireless protocols in IoT (e.g., Institute of Electrical and Electronics Engineers (IEEE) 802.15.4 standard in the 2.4 GHz band demands about 2.1 GMACS. It is worth noting that the value of 2.1 GMACS for IEEE 802.15.4 is calculated by recompiling the works implemented in [6, 7].

In this chapter, to our best knowledge, we introduce for the first time a complete multi-channel TRX capable of both transmitting and receiving the data on multiple channels concurrently. We have applied Hardware Virtualization (HV) in our multi-channel TRX prototyping by fully exploiting the high processing capability of a modern SDR platform. HV is a technology that allows users to create multiple logical instances from a single physical instance implemented on HW. It enables us to use the same HW of single channel TRX for multiple channels with only a limited amount of extra logic added to manage the virtualization overhead.

To this end, we have prototyped the 8 $\times$ 8 multi-channel TRX on an SDR in two approaches:

- **The Hardware Duplication (HD) approach**, in which we have simply duplicated the HW of a single channel IEEE 802.15.4 compliant TRX for multiple channels on the SDR. The HD approach is implemented as a benchmark to evaluate the proposed approach.
- **Our proposed HV approach**, in which we have used the same HW of single channel TRX for multiple channels, with some extra logic added to manage the virtualization overhead.

By prototyping and comparing these two approaches, it is validated that the HV approach is not only efficient in terms of hardware utilization but also provides the same performance as the HD approach.

2.2 Related Work

In the first section, SDR based multi-channel solutions presented in the literature are investigated. Next, commercially available state-of-the-art chipset based solutions are discussed.

2.2.1 SDR based Solutions

An IEEE 802.15.4 multi-channel Receiver is described in [3]. This solution uses USRP2 [8] to capture the packets of 5 contiguous channels located in the 2.4 GHz ISM band. Further, a dedicated Digital Down Converter (DDC) and demodulator is implemented for each channel in GNU Radio [9]. The major downside of the solution is that it allocates a dedicated DDC and demodulator for each channel. The duplication of these processing modules increases the load on a Central Processing Unit (CPU), which can cause samples of USRP to overflow and packets not being decoded. Yohe and co-authors [10] have presented Multi-Protocol Access Point (MPAP), an HV architecture for heterogeneous wireless Access Points (APs). In this work, an 802.11 g and two 802.15.4 radio receivers are integrated on the Sora platform [11], where IQ samples received by wide-band Radio Frequency (RF) front-end are fed into multi-core Personal Computer (PC). The PHY and upper layers of the respective standards are running on the PC. The incorporation of the PC not only makes the platform inappropriate for the solutions acquiring ultra-low latency (e.g., factory automation) but also causes it to be costly and bulky. Jiao et al. [12] has applied radio HV on System on Chip (SoC), where DDC banks, related PHY layers are implemented in FPGA, and corresponding scheduling software is running on an embedded processing unit. The authors claim that the design can decode 2 Wireless Fidelity (Wi-Fi) and 8 IEEE 802.15.4

channels concurrently, but the implementation is limited to preamble detection. In other words, it does not decode the complete packets of the protocols.

2.2.2 Commercial Radio Transceiver based Solutions

A multi-channel packet sniffer is developed in [4], where multiple IEEE 802.15.4 based commercial off-the-shelf radio are adopted to capture the packet. The architecture inherits two disadvantages: (1) it employs a dedicated radio device for a channel and all the devices have their own clock source, leading to the addition of a time synchronizer to overcome clock drifting, and (2) non-compact, because the developer has to use a separate commercial radio for each channel. Similarly, a special probe has been introduced for multi-channel packet sniffer in [13]. The probe is composed of an FPGA to act as a supervisor, and 15 IEEE 802.15.4 compliant commercial radios. In order to maintain the synchronization among the radio radios, a common clock source is implemented in FPGA. The probe has fixed the clock drifting issue, but it still have the above-mentioned non-compact issue. A multi-channel Wi-Fi sniffer able to decode multiple consecutive channels in 2.4 GHz and 5 GHz is designed by Pradeep and his colleagues [14]. They have also employed a dedicated commercial chip for each channel.

In summary, there exists two categories of related work for our multi-channel TRX, i.e. the solutions based on SDR and the solutions based on commercial chipsets. In both categories, there is no candidate capable of multi-channel transmission, all solutions are receiver only. The receivers on multiple channels are achieved by duplicating software/hardware modules, leading to (i) the consumption of extra resources, (ii) the increase of cost and form factor (in the case of commercial chipset based solution), and (iii) possibly extra complications such as the need to overcome clock drifting across multiple hardware platforms. The novelties of our solution are following:

- Our solution Concurrent Multi-Channel Virtual Transceiver (CMCVT) consists of a single PHY implemented on FPGA of an SDR. It is capable of both transmit and reciver.

- The concurrent multi-channel operation is achieved by HV, taking full advantage of the high processing capabilities of the modern SDR.

- Lastly, because the CMCVT is working on a single device, it does not have the clock drifting/synchronization issue.

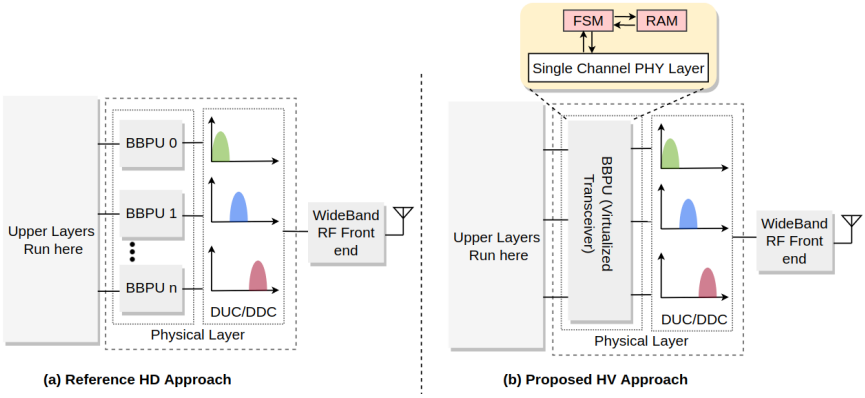


Figure 2.1: A block diagram showing a general comparison between (a) the duplication approach to implement the multi-channel transceiver and (b) the proposed HV approach to implement the multi-channel transceiver.

2.3 SDR based Concurrent Multi-Channel Virtual Transceiver

A multi-channel TRX obtained by using HD approach is referred to as Conventional Multi-Channel Transceiver (CMC-TRX) and is used as a reference for comparison; while our proposed CMCVT is achieved by using the HV approach. Fig. 2.1 highlights a general comparison between the reference HD approach and our HV approach. The common components shared in both approaches are wideband RF front-end, upper layers running on a processor, Digital Up Converter (DUC) which converts the baseband signal to some Intermediate Frequency (IF) signal, and Digital Down Converter (DDC) which converts IF signal back to the baseband signal. The main differences between the HD and the HV based architectures is the implementation of Baseband Processing Unit (BBPU) of the PHY layer. As depicted in Fig. 2.1 (a), the HD approach assigns a separate BBPU to each channel, while in HV approach, the same BBPU is timely shared among multiple channels. We apply the overclocking concept in the HV approach: the PHY layer is running at $N \times CLK_{bb}$, where N and CLK_{bb} represent the number of channels and baseband data rate of a wireless standard (e.g., 250 kbps is the CLK_{bb} in IEEE 802.15.4 standard), respectively.

Fig. 2.1 (b) elaborates the internal block diagram of HV approach for CMCVT (see yellow box in Fig. 2.1). We propose three simple but effective steps to convert any single channel TRX into HV based multi-channel TRX: (1) obtain a single channel PHY layer of any wireless standard. For this chapter, we have recompiled and modified the existing single channel implementation [6, 7] written in Hardware Descriptive Language (HDL) of IEEE 802.15.4 standard [15]. Without the

loss of generality, among the three different PHY modes (i.e., 20 kbps, 40 kbps, 250 kbps) specified in the standard, we have chosen the PHY layer of data rate 250 kbps functioning in 2.4 GHz band for the implementation. It is worth mentioning that the chosen PHY layer uses Offset Quadrature Phase Shift Keying (O-QPSK) as a modulation scheme, (2) Identify and expose the context saving and restoring signals involved during context switching of the single channel PHY layer. Context switching is a commonly used technique in CPU domain in which context or states of a process are stored so that they can be restored at a time when CPU resumes the execution of the process. The context switching is a feature of multitasking, which enables a CPU to be shared by multiple processes. (3) Implement a Context Switching Finite State Machine (CS-FSM) for the context switching. The Finite State Machine (FSM) is the most important part of our HV approach. It is responsible for the correct functioning of the CMCVT. These three simple steps enable us to convert CMCVT from a conceptual idea into a real-life solution. It is obvious that each CMCVT's PHY layer can be further decoupled into transmitter's PHY and receiver's PHY layers, both are detailed in the following sections.

2.3.1 Physical Layer of the Multi-Channel Virtual Transmitter

A detailed diagram of the Multi-Channel Virtual Transmitter (MCVT) is elaborated in Fig. 2.2. The PHY layer of MCVT implemented in FPGA is composed of DUC bank, and BBPU.

2.3.1.1 The Working Flow of the Multi-Channel Virtual Transmitter's PHY layer

The working flow of the MCVT is as follow:

- The Media Access Control (MAC) layer running on the embedded Processor System (PS) configures the BBPU and the RF front-end (see control path in Fig. 2.2). For instance, the sampling frequency, bandwidth of the RF front-end, and the number of potential channels on which data is to be transmitted in BBPU are amongst the configurable parameters.

- Then, the MAC layer starts sending data to BBPU through Direct Memory Access (DMA) (see data path in Fig. 2.2). The BBPU keeps storing the data into Random Access Memory (RAM), until it receives the corresponding data of all the channel(s) defined in step (1).

- Lastly, the BBPU initiates the transmission, and by generating interrupts, it updates the MAC layer about the status of transmission.

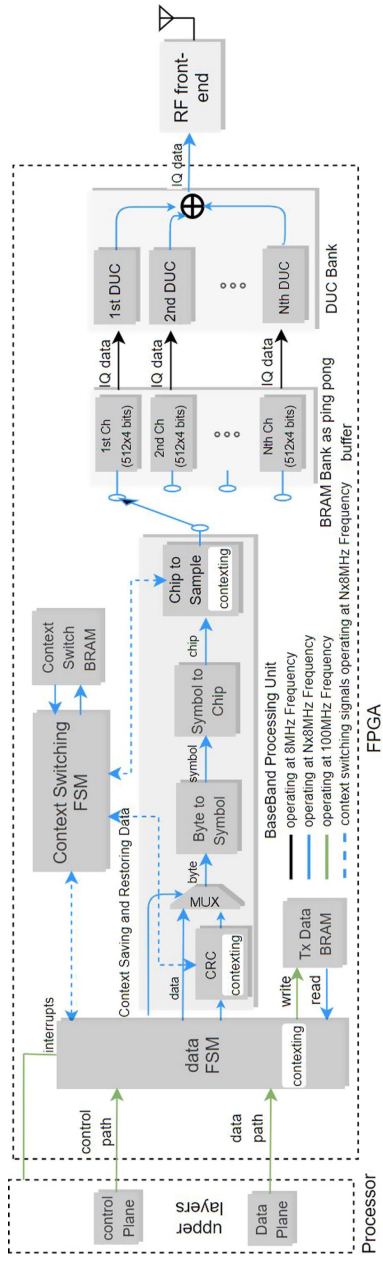


Figure 2.2: A detailed diagram showing all the modules involved in realization of the MCVT.

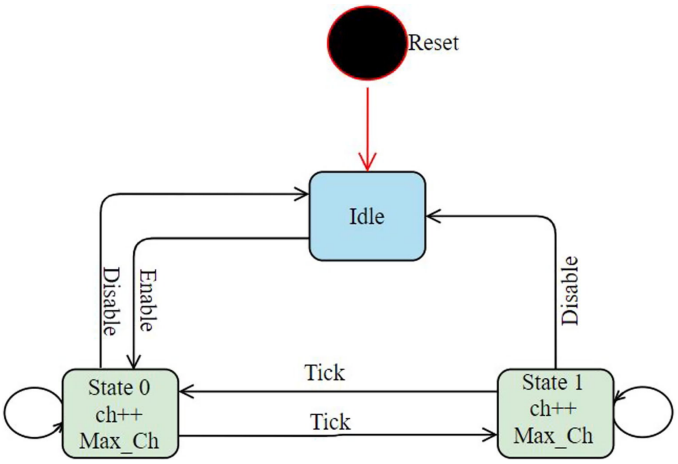


Figure 2.3: The state diagram of context switching finite state machine.

2.3.1.2 Implementation of the Multi-Channel Virtual Transmitter's PHY layer

The HV is applied for the BBPU. Instead of allocating a separate BBPU for each channel, the HV allows to utilize the same BBPU for multiple channels. We leverage expertise in multitasking, pipelining, and multi-clock domains to achieve virtualization of the BBPU. The technologies are detailed in the following sub-sections.

Multitasking Similar to multitasking in CPU, our design also inherits the context storing-restoring problem. In principle, any functional block/program that has delay, memory, pipeline, and internal states in multitasking requires context saving and restoring operation. To this end, we introduce a CS-FSM and Block Random Access Memory (BRAM) in our design (as shown in Fig. 2.2). In the BBPU, we have identified that only three modules require context switching, namely: data FSM (due to internal states), Cyclic Redundancy Check (CRC) (due to memory), and chip to sample (due to delay). After each processing time unit, hereafter referred to as a tick, the context of these modules needs to be saved and the context for next channel needs to be restored. The CS-FSM is responsible for all these operations. Fig. 2.3 shows the state diagram of the FSM. At the beginning, the FSM enters into the first state (indicated by state 0 in the Fig. 2.3) when an 'Enable' signal is high. Then, the FSM instructs the BBPU to start processing the data, meanwhile it enables corresponding BRAM in the BRAM bank (see Fig. 2.2)) to read/write the IQ samples. After a tick, the FSM enters into second state (indicated by state 1 in Fig. 2.3) and during the transition, it stores states of the current channel and restores states of the next channel. The FSM keeps the record of

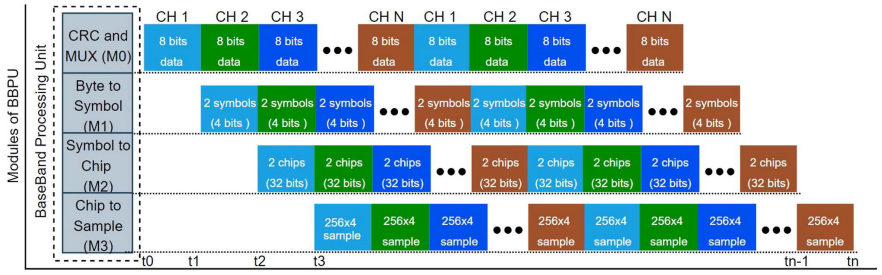


Figure 2.4: Applying pipelining in the Multi-Channel Virtual Transmitter.

the current and next channel number in an 8 bit register (indicated by 'ch' in the Fig. 2.3). When channel number reaches maximum supported channels (indicated by 'Max_ch' in the Fig. 2.3), FSM is wrapped around to the first channel. Instead of using a dedicated state for each channel, the FSM always has three states, irrespective to the number of channels that the MCVT supports. This FSM design ensures the correct operation of the MCVT and at the same time offers high efficiency in terms of HW utilization.

Pipelining Taking advantage from the modular structure, we have further included pipelining in the BBPU. The pipelining helps to reduce the clock overhead caused by context saving and restoring operations. An example of pipelining for n channels is illustrated in Fig. 2.4, where horizontal axis represents the time and vertical axis reflects the modules of the BBPU. At t0 time, CRC and multiplexer module (indicated by M0 in Fig. 2.4) begins processing the data of First Channel (CH1). After a tick (i.e., the time equivalent of processing 256 IQ samples), M0 switch to CH2, meanwhile the byte to symbol module (indicated by M1 in Fig. 2.4) is enabled for CH1. During the third tick (i.e., after t3), all the modules are busy in processing the data of consecutive channels. The output of the chip to sample(indicated by M3 in Fig. 2.4) module is stored into BRAM, which decouples the BBPU from the DUC bank. M3 generates 256 symbols after each tick. There is a separate BRAM for each channel with each BRAM has the capacity to store 512 symbols. The BRAM behaves as a ping pong buffer, which means the RAM can accept new symbols in the 257–512 locations while sending out symbols in the first 256 locations, and vice versa. This configuration prevents the DUC of a channel from processing the wrong IQ symbols.

Multi-clock domains To meet the critical constraints of the specific standard, the MCVT uses three different clocks as indicated in Fig. 2.2: (1) the clock specified for control and data paths, which is 100 MHz in our design (2) the clock at which BBPU is operation, running at $N \times CLK_{bb}$ (CLK_{bb} is 8 MHz in our

design), and (3) the clock at which each DUC in DUC bank reads the data from ping pong BRAM at CLK_{bb} rate. Due to multi-clock domains, the MCVT is prone to metastability. To mitigate the metastability situation, we have introduced 2-flop synchronizer and dual port BRAM for single-bit, and multi-bit data signals, respectively.

2.3.2 Physical Layer of the Multi-Channel Virtual Receiver

Unlike MCVT, turning the Multi-Channel Virtual Receiver (MCVR) from conceptual idea into a working solution is way more challenging. Almost all the modules constituting the BBPU of MCVR needs context saving and restoring, adding extra complexity in the architecture. A detailed diagram of the MCVR is presented in Fig. 2.5, where the corresponding PHY layer incorporates BBPU and DDC bank is realized in FPGA.

2.3.2.1 The Working Flow of the Multi-Channel Virtual Receiver's PHY Layer

The working flow of the MCVR is as follows:

- After the configuration of PHY layer and RF front-end by the upper layer (indicated by control plane in Fig. 2.5), the DDCs of corresponding channels in DDC bank first shift the center frequency of IQ samples, down sample and then write the incoming samples to the BRAMs.
- The BBPU begins to decode the samples read from the first BRAM ping pong buffer.
- After a tick (i.e., the time equivalent of reading 8 IQ samples), the "sample reading multiplexer" switches to the second BRAM, meanwhile CS-FSM concurrently performs context saving for the current channel and restoring for the subsequent channel.
- In order for the upper layer to recognize to which channel an incoming packet belongs to, the BBPU appends one extra byte to the decoded data representing the channel number.

2.3.2.2 Implementation of the Multi-Channel Virtual Receiver's PHY Layer

Similar to MCVT, the HV is only applied on BBPU of MCVR. Likewise, the MCVR has benefitted from multi-clock domain, pipelining and multitasking, which together alleviate the clocks overhead added during context saving and restoring. All these technologies are already thoroughly explained in MCVT's implementation section. In addition to the implementation of MCVR, it requires an Automatic

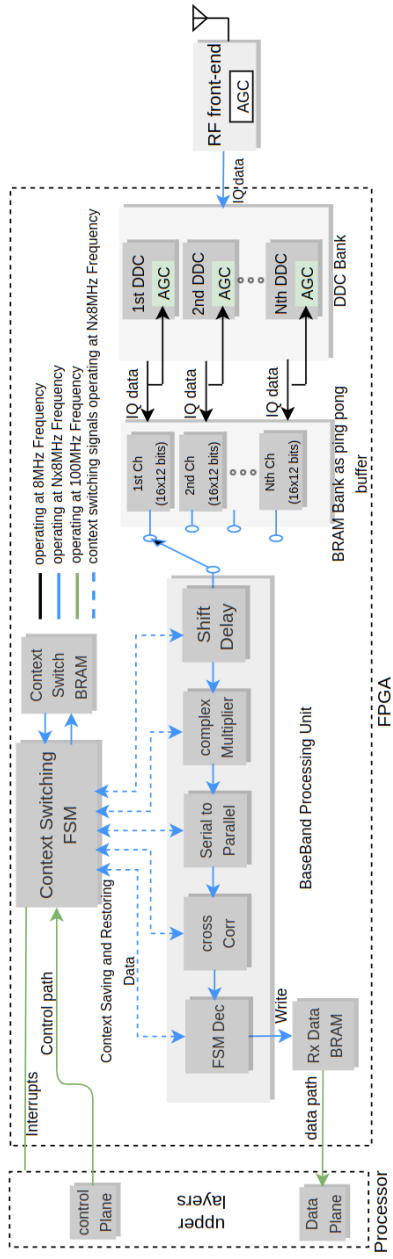


Figure 2.5: A detailed diagram showing all the modules involved in realization of the MCVR.

Table 2.1: A comparison of hardware utilization efficiency of CMCR and our MCVR in term of logic consumption.

Channels	Resource	CMCR	MCVR	Improved Efficiency
1	LUTs	854	854	0
	FFs	1073	1073	0
2	LUTs	1708	2128	-24.59
	FFs	2146	1412	34.20
4	LUTs	3416	2168	36.53
	FFs	4292	1432	66.64
8	LUTs	6832	2220	67.51
	FFs	8584	1464	82.95

Gain Controller (AGC) for each channel that tunes the signal strength to compensate for channel-specific losses and fading required for proper decoding. Since the analog AGC (see RF front-end in Fig. 2.5) of the used RF front-end works on the wideband RF signal, it is observed that it fails to provide enough signal strength for each channel required to accurately decode it. The described issue is mitigated by implementing a dedicated digital AGC for each channel (see DDC bank in Fig. 2.5), which together with analog AGC enables the MCVR to correctly decode the data in all possible situations.

2.4 Results and Discussions

We have combined the MCVT and MCVR into the CMCVT, in which the corresponding BBPUs can independently transmit/receive data on multiple channels. The SDR chosen in the particular setup is composed of ZedBoard [16] and FMCOMMS2 board [17]. ZedBoard is a low-cost development board for the Xilinx Zynq 7000 SoC [18] and the SoC is further comprised of Programmable Logic (PL) (an alternative term for FPGA) and PS. The PHY layers of CMCVT are implemented in the PL part, while FMCOMMS2 board is used as analog RF front-end. A single channel TRX proposed in [6] is used as a building block to implement a multi-channel TRX. Since the sampling rate of the TRX in [6] is 8 MHz and maximum sampling rate supported by FMCOMMS2 is 64 MHz, the implementation of CMCVT is restricted to 8 parallel channels. Similarly, the implementation uses the same Digital-to-Analog Converter (DAC) in MCVT multi-channels decreasing the transmission power on each channel.

The CMC-TRX, which is used as a benchmark for comparison, is realized by duplicating the single channel TRX in [6]. Like every design in FPGA, our BBPUs have logic and memory parts. The logic part of a design is mapped on LUT and FF, whereas the memory part is placed on RAM. The comparison between HW

Table 2.2: A comparison of hardware utilization efficiency of CMCT and our MCVT in term of logic consumption.

Channels	Resource	CMCT	MCVT	Improved Efficiency
1	LUTs	272	272	0
	FFs	442	442	0
2	LUTs	544	532	2.21
	FFs	884	565	36.09
4	LUTs	1088	619	43.11
	FFs	1768	593	66.46
8	LUTs	2176	739	66.04
	FFs	3536	641	81.87

utilization efficiencies of CMCVT and CMC-TRX is performed in terms logic and memory consumption. Instead of directly comparing the HW utilization of CMCVT with CMC-TRX, MCVT and MCVR are separately compared against their corresponding conventional counterparts. The efficiency shown is calculated by using (2.1); it represents the improvement in HW resource utilization of CMCVT relative to CMC-TRX.

$$ImprovedEfficiency = \left(\frac{HW_{conv} - HW_{our}}{HW_{conv}}\right) \times 100 \qquad (2.1)$$

where HW_{conv} and HW_{our} represents the HW consumed by the duplication and our approaches, respectively, and HW can be LUT, FF or RAM.

2.4.1 Evaluation of Logic Consumption

Table 2.1 depicts the comparison of HW utilization efficiency of our MCVR against Conventional Multi-Channel Receiver (CMCR) obtained by using HD approach, while Table 2.2 illustrates the comparison of our MCVT against Conventional Multi-Channel Transmitter (CMCT). It is noteworthy that the tables only contain LUTs and FFs (i.e., they only include logic parts of the respective designs). It can be seen from Table 2.1, Table 2.2 that the benefit of the proposed HV approach is more pronounced for higher number of parallel channels. Under the setting of 2 channels, our approach interestingly provides either negative (in MCVR case) or slightly improved (in MCVT case) efficiency. The degraded efficiency for the particular case is expected, because our approach involves context storing-restoring, which requires extra logic. However, most part of the extra logic do not change as the number of parallel channels increases, resulting in more improvement in the HW utilization efficiency for higher number of channels. It is logical that the best case is achieved for 8 channels, where MCVR saves 82.95% FFs and 67.51%

Table 2.3: A comparison of BRAM utilization of CMCR, MCVR, CMCT, and MCVT.

Channels	Resource	CMCR	MCVR	CMCT	MCVT
1	BRAMs	0.5	0.5	0.5	0.5
2	BRAMs	1	2	1	2
4	BRAMs	2	3	2	3
8	BRAMs	4	5	4	5

LUTs as compared to CMCR (shown in Table 2.1), and MCVT reduces 81.87% FFs and 66.04% LUTs as compared to CMCT (shown in Table 2.2).

2.4.2 Evaluation of Memory Consumption

The PL part incorporates two types of RAMs; Distributed RAM (DRAM) and BRAM. LUTs can be configured as either logic or DRAM. In contrast to DRAM, BRAMs are dedicated RAMs and are located at fixed positions in FPGA. It is generally recommended to map memory part of the design on BRAM, so that more LUTs and FFs would be available for logic implementations. We therefore have mapped the memory parts of the designs onto BRAM. The BRAMs consumed in the design for different number of channels are summarized in Table 2.3. There are 140 BRAMs of size 36 Kb in ZedBoard FPGA, and the numbers in Table 2.3 represent the numbers of utilized 36 Kb BRAMs. In contrast to CMC-TRX that employs a single memory (as shown the data BRAM in Fig. 2.2, Fig. 2.5) to read-/store the packets, the CMCVT includes two extra memories; a context switching BRAM to store and restore the states of corresponding modules of the BBPU every time the BBPU switches to a new channel, and a Ping Pong BRAM to store the incoming/outgoing samples for further processing. Although these extra BRAMs potentially deteriorate the memory utilization efficiency of our approach, the increment is not significant. Our approach uses at most one extra BRAM than the duplication approach, as shown in Table 2.3.

2.4.3 Multi-channel Receiver Sensitivity Measurement and Analysis

Fig. 2.6 shows the magnitude response of Finite Impulse Response (FIR) used in the DDC of the CMCVT. It can be seen from the Fig. 2.6 that the FIR provides an attenuation for more than -33 dB after 5 MHz. The IEEE 802.15.4 standard defines a channel spacing of a 5 MHz among the adjacent channels when operating in 2.4 GHz band. Further, the values of adjacent and alternate channel interference rejection defined in the standard are 0 dB and 30 dB, respectively. Thus, the FIR provides the required adjacent channel interference rejection to our receiver dur-

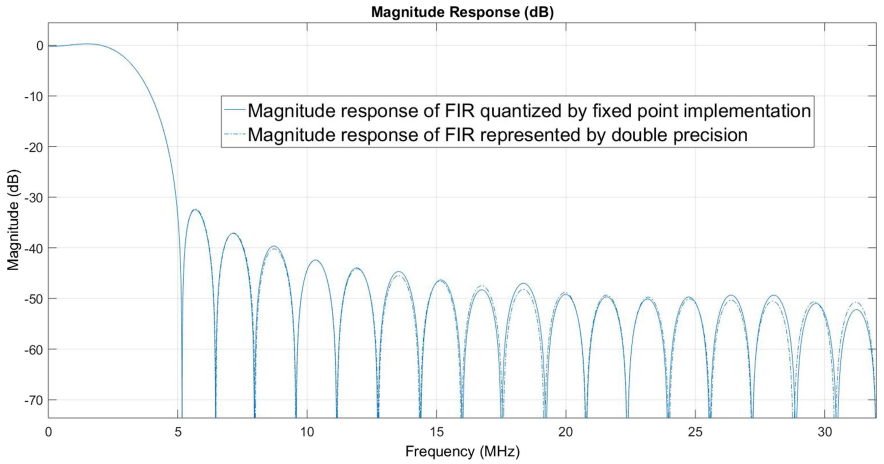


Figure 2.6: Magnitude response of FIR used in DDC of the CMCVT. Where, double precision and fixed point magnitude responses are used as a reference and in the implementation of CMCVT, respectively.

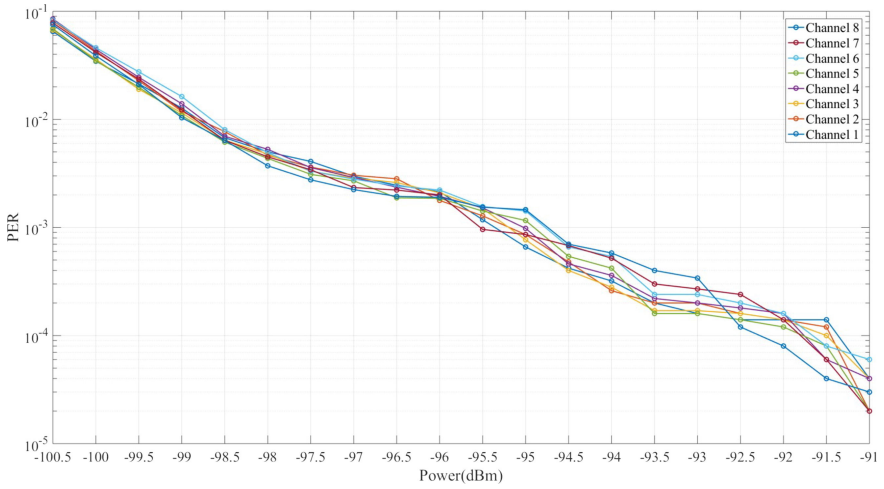


Figure 2.7: Sensitivity measurements on all 8 channels of the multi-channel virtual receiver.

ing decoding the data from multiple channels simultaneously. Moreover, we have measured the receiver's sensitivity according to the requirements defined in IEEE 802.15.4 standard [15]. The input power at which the Packet Error Rate (PER) drops to 1% is termed as the sensitivity of a receiver. We have transmitted 50,000 packets with each containing 20 bytes in the air. The original implementation [6] has a receiver sensitivity of -98.5 dBm and we have implemented 8 concurrent

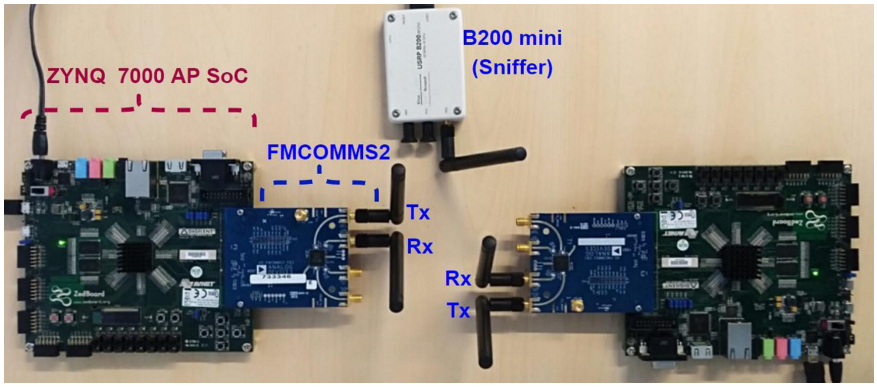


Figure 2.8: An experimental setup for the real time validation of CMCVT.

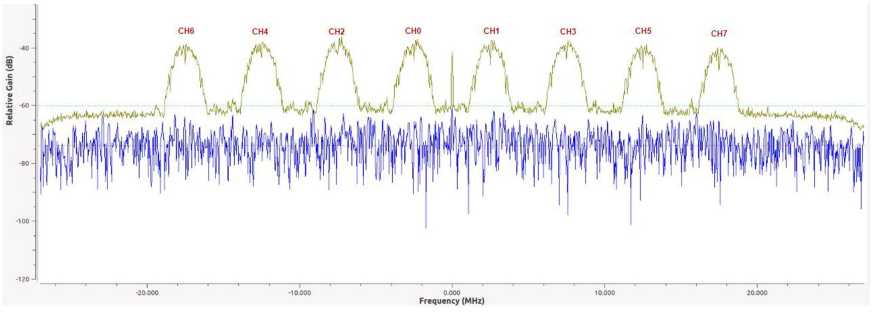


Figure 2.9: Power spectrum view captured by a USRP B200 mini when the SDR (i.e., in MCVT mode) is sending packets on all 8 channels concurrently.

channels in MCVR by modifying the existing single channel receiver. We thus expect the same sensitivity values on all the 8 channels of the MCVR. To this end, we have measured the PER for each channel under a range of received power (in dBm). It can be seen in Fig. 2.7 that all 8 channels of the MCVR have the same sensitivity value which is -98.5 dBm. Fig. 2.7 however shows a minor difference of PERs among different channels. It is because the measurements are done on a real-life setup, and it is hard to achieve the same PERs even for the same channel[1].

[1]Note that the PERs' difference among channels become more visible for low PER, it is because 50,0000 packets are used for PER measurements, which are not sufficient for low PER. However, they are sufficient to quantify the receiver sensitivity of MCVR.

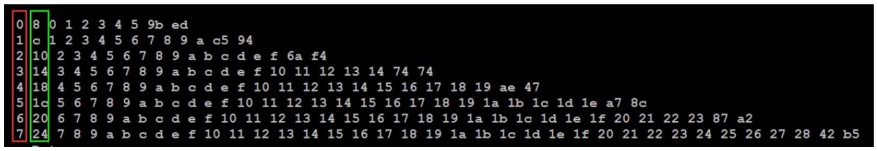


Figure 2.10: Packets decoded by the SDR when it is receiving the packets from 8 consecutive channels.

2.4.4 Proof of Concept Experiment

After discussing the HW utilization and sensitivity results of the CMCVT, an experiment is performed to validate the virtual TRX. To this end, two SDRs are used, each consisting of a ZedBoard and an FMCOMMS2 (see Fig. 2.8). A USRP B200 mini [8] is used to visualize the wideband power spectrum when CMCVT is transmitting concurrently on 8 channels. The power spectrum view is shown in Fig. 2.9, where the envelope of 8 signals are clearly visible. It shows that CMCVT is capable of transmitting data on multiple channels concurrently. Fig. 2.10 has displayed the packets decoded by the other SDR. As explained before, BBPU appends one extra byte to the decoded packets to indicate the channel number (red box in Fig. 2.10). The second byte indicates the packet length as is defined in the standard (green box in Fig. 2.10), and the rest are the payload of PHY layer.

2.5 Conclusions

This chapter introduces a multi-channel virtual TRX architecture, leveraging the concept of HV and the high processing capability of FPGAs or ASICs in modern communication devices. Instead of allocating a dedicated TRX for each radio channel, our TRX creates multiple logical TRXs based on a single physical TRX. Such a design and implementation is highly efficient in terms of hardware utilization, hence effectively reducing the silicon footprint in ASIC design. To validate this new concept, we have prototyped an IEEE 802.15.4 complaint multi-channel virtual TRX on an SDR. The virtual TRX behaves as multiple dedicated TRXs, which can independently transmit/receive data on up to 8 channels. Experimental evaluation of our virtual TRX reveals that it holds the same performance as the multiple physical TRXs on different channels, but 82.63% FFs and 67.15% LUTs hardware resources can be saved for the particular case of 8 channels.

Our proposed method is generic, easy to implement and can therefore be applied on any existing or future wireless standards in IoT domain. Any single channel TRX can be easily virtualized by performing the following 2 steps: (1) identify the context saving and restoring signals from the original design, (2) add an extra FSM and RAM for context storing and restoring operations. We have verified our

HV concept in a single antenna scenario, but also other emerging wireless technologies can benefit from hardware virtualization, such as non-orthogonal multiple access and massive multiple-input multiple-output. As virtualization is entirely applied on the digital side of a transceiver, it is fully transparent to the RF part. In the future, we are planning to extend the proposed method for virtualization towards other wireless standards, and also the upper layers of the communication stack.

Acknowledgment

The project leading to this publication has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 732174 (ORCA project).

References

[1] P. Desai, A. Sheth, and P. Anantharam. *Semantic gateway as a service architecture for iot interoperability*. In 2015 IEEE International Conference on Mobile Services, pages 313–319. IEEE, 2015. doi:https://doi.org/10.1109/MobServ.2015.51.

[2] M. v. S. Antonius de Graauw. *Concurrent multiband transceiver*, U.S. Patent US9106314B2, Aug. 2015.

[3] L. Choong. *Multi-channel IEEE 802.15.4 packet capture using software defined radio*. UCLA Networked & Embedded Sensing Lab, 3:1–20, 2009.

[4] S.-e. Yoo, P. K. Chong, J. Bae, T.-S. Kim, H. Kim, and J. Yoo. *Multi-channel packet-analysis system based on IEEE 802.15.4 packet-capturing modules*. International Journal of Distributed Sensor Networks, 10(9):216504, 2014. doi:https://doi.org/10.1155/2014/216504.

[5] *A user guide for 7 Series FPGAs Configurable Logic Block*, 2016. Available from: https://www.xilinx.com/support/documentation/user_guides/ug474_7Series_CLB.pdf.

[6] M. Aslam, X. Jiao, W. Liu, and I. Moerman. *An approach to achieve zero turnaround time in TDD operation on SDR front-end*. IEEE Access, 6:75461–75470, 2018. doi:https://doi.org/10.1109/ACCESS.2018.2883253.

[7] T. Kazaz, X. Jiao, M. Kulin, and I. Moerman. *WiSCoP-Wireless Sensor Communication Prototyping Platform*. arXiv preprint arXiv:1612.02900, 2016. doi:https://doi.org/10.48550/arXiv.1612.02900.

[8] *The Universal Software Radio Peripheral (USRP)*, 2022. Available from: https://www.ettus.com/.

[9] *The free and open-source software development toolkit*, 2022. Available from: https://www.gnuradio.org/.

[10] Y. He, J. Fang, J. Zhang, H. Shen, K. Tan, and Y. Zhang. *Mpap: virtualization architecture for heterogenous wireless aps*. ACM SIGCOMM Computer Communication Review, 40(4):475–476, 2010. doi:https://doi.org/10.1145/1851275.1851271.

[11] *Microsoft Research Software Radio (Sora)*, 2022. Available from: https://www.microsoft.com/en-us/research/project/microsoft-research-software-radio-sora/.

[12] X. Jiao, I. Moerman, W. Liu, and F. A. P. d. Figueiredo. *Radio hardware virtualization for coping with dynamic heterogeneous wireless environments*. In International Conference on Cognitive Radio Oriented Wireless Networks, pages 287–297. Springer, 2017. doi:https://doi.org/10.1007/978-3-319-76207-4_24.

[13] P. Ferrari, A. Flammini, D. Marioli, S. Rinaldi, and E. Sisinni. *On the implementation and performance assessment of a wirelessHART distributed packet analyzer*. IEEE Transactions on Instrumentation and Measurement, 59(5):1342–1352, 2010. doi:https://doi.org/10.1109/TIM.2010.2040907.

[14] P. Reddy, H. Sharma, and D. Paulraj. *Multi Channel Wi-Fi Sniffer*. In 2008 4th International Conference on Wireless Communications, Networking and Mobile Computing, pages 1–6. IEEE, 2008. doi:https://doi.org/10.1109/WiCom.2008.727.

[15] *IEEE Standard for Low-Rate Wireless Networks*, 2020. Available from: https://standards.ieee.org/ieee/802.15.4/7029/.

[16] *ZedBoard, A development board for Zynq-7000 SoC*, 2022. Available from: https://www.xilinx.com/products/boards-and-kits/1-8dyf-11.html.

[17] *User Guide of AD-FMCOMMS2-EBZ an FMC Board for the AD9361*, 2022. Available from: https://wiki.analog.com/resources/eval/user-guides/ad-fmcomms2-ebz.

[18] *An overview of Zynq-7000 SoC Data Sheet*, 2022. Available from: https://www.xilinx.com/support/documentation/data_sheets/ds190-Zynq-7000-Overview.pdf.

3

A Novel Hardware Efficient Design for IEEE 802.11ax compliant OFDMA Transceiver using Hardware Virtualization

The Multi User (MU) communication can be broadly categorized into two types. In the first type of MU communication such as concurrent multi-channel communication and MU-Multiple Input Multiple Output (MIMO), the capacity of the system increases with the number of users in MU communication. Chapter 2 is an example of such a MU communication. In the second type of MU communication such as MU-Orthogonal Frequency Division Multiple Access (OFDMA), the capacity of the system remains fixed irrespective of the number of users in MU communication. This chapter proposes hardware efficient design for an IEEE 802.11ax compliant MU OFDMA Transceiver (TRX).

⋆ ⋆ ⋆

Muhammad Aslam, Xianjun Jiao, Wei Liu, Michael Mehari, Thijs Havinga, and Ingrid Moerman.

Submitted to The 28th Annual International Conference On Mobile Computing And Networking Hyatt Regency Sydney, Australia, Oct. 2022.

Abstract The introduction of Multi User (MU) communication in IEEE 802.11ax in the frequency domain via MU-Orthogonal Frequency Division Multiple Access (OFDMA) and in the spatial domain via MU-Multiple Input Multiple Output (MIMO) enables the Access Point (AP) to serve up to 128 Stations (STAs) in a group or schedule. However, the MU functionality poses new challenges in the chip design. Existing MU Transceivers (TRXs) rely on the duplication approach wherein a dedicated hardware is utilized per user. The hardware footprint of such an approach increases proportionally with the number of users simultaneously served in the MU TRX. The inefficiency in duplication approach wastes hardware resources and underutilizes potentials of modern silicon solutions, such as a Field Programmable Gate Arrays (FPGA) in a Software Defined Radio (SDR) or Application Specific Integrated Chip (ASIC) in COTS devices.

This paper introduces a novel hardware efficient design for an MU TRX. Unlike the duplication approach, the proposed design for the MU TRX has comparable hardware footprint of a single User (SU) TRX regardless of the number of users being served, thanks to the hardware virtualization technique. The applicability of the design is initially validated for IEEE 802.11ax compliant MU-OFDMA transmitter on an FPGA of a modern SDR. The performance and hardware consumption is compared against the conventional duplication approach. This first demonstration supports up to 4 STAs to transmit data simultaneously, but is extendable up to 74, the maximum number of STA supported in MU-OFDMA for IEEE 802.11ax standard. The experimental results show that the hardware virtualization based MU-OFDMA transmitter provides the same performance and consumes $\approx$3 times less look up tables and 2.57 times less Registers in comparison with the conventional duplication approach.

3.1 Introduction

Nowadays, wireless internet access is ubiquitous and is bringing unprecedented convenience to almost every aspect of our life. Institute of Electrical and Electronics Engineers (IEEE) 802.11 Wireless Local Area Network (WLAN) standard, commercially referred to as Wireless Fidelity (Wi-Fi), holds a dominant position in providing Internet access. In addition to Internet access, WLAN is also increasingly used in other existing and emerging applications including Industry 4.0, Internet of Things (IoT), Industrial IoT (IIoT), Internet of Everything (IoE) [1]. According to the Wi-Fi Alliance, the number of operational Wi-Fi devices will reach nearly 18 billion in 2022 [2]. In order to cater the constantly increased requirement of these applications, the WLAN standard is continually evolving and adding new technologies and features to the current standard.

The evolution of the first five WLAN standards (i.e., IEEE 802.11a/b/g/n/ac) was mainly revolved around increasing data rate achieved by exploiting many

state-of-the-art technologies [3]. For instance, IEEE 802.11a/g increased the data rate up to 54 Mbps by exploiting Orthogonal Frequency Division Multiplexing (OFDM) and Single Input Single Output (SISO) antenna technology. OFDM is a modulation technique used to mitigate inter symbol interference caused by multi-path propagation and narrowband interference. IEEE 802.11n and IEEE 802.11ac achieved data rates up to 600 Mbps and 6.9 Gbps, respectively, primarily by introducing MIMO, increasing channel bandwidth and enhanced modulation and coding rates. MIMO is an antenna technology used to enhance either the throughput through spatial multiplexing or the reliability through spatial diversity. All of these standards however can only service a SU at a time, meaning that the wireless medium can only be occupied by a single STA either transmitting to or receiving from an AP. Only when IEEE 802.11ac introduced the Down Link (DL) Multi User (MU) MIMO in the second wave, an AP was able to transmit to multiple STAs simultaneously.

The IEEE 802.11ax standard [4] (also known as High Efficiency (HE) WLAN), for the first time, introduced both DL and Up Link (UL) MU transmission. The main objective of the standard is not only to increase data rate, but also to improve spectral efficiency especially in high-density public environments, like trains, stadiums and airports. The key enablers for the MU transmission are OFDMA and MU-MIMO. OFDMA is the MU version of OFDM which allows simultaneous MU transmission in the same frequency band by assigning subsets of Subcarriers (SCs) named Resource Unit (RU) to individual users. Though MU communication is recently introduced in IEEE 802.11ax, it already exists in other standards such as IEEE 802.16 (Worldwide Interoperability for Microwave Access (WiMAX)), Long Term Evolution (LTE), 5$^{\text{th}}$ Generation (5G) New Radio (NR) [5, 6]. On the one hand, the MU communication has tremendous potential to improve the Wi-Fi performance in spectrum and spatial domains by exploiting MU-OFDMA and MU-MIMO, respectively. On the other hand, it also requires a complex Hardware (HW) design for the radio TRX to realize these sophisticated techniques. In other words, more HW resources are required to implement such a complex HW design. Existing approaches [7, 8] employ a Hardware Duplication (HD) approach to design the Physical (PHY) layer of MU TRX (i.e., dedicated HW resources are assigned for each user). With such approaches, the consumption of the HW resources increases proportionally with the number of users in MU TRX. Consequently, a MU TRX realised by an HD approach occupies a larger die area of a chip, making it more expensive. The amount of resources consumed in an HD approach can be formulated using (3.1),

$$HW_{MU_{HD}} = N_{users} \times HW_{SU} \tag{3.1}$$

where, $HW_{MU_{HD}}$ reflects the total amount of HW consumed by the PHY layer of an MU TRX using HD approach, N_{users} denotes the number of users and HW_{SU}

represents the HW consumed by the PHY layer of an SU TRX. In the worst case, MU-OFDMA TRX in IEEE 802.11ax supports up to 74 simultaneous users, meaning that an HD approach will consume 74 $\times$ HW_{SU} resources. For this reason, an alternative, more hardware-efficient design for the PHY layer of MU TRX is highly desired, which is the focus of this paper.

Ideally, a HW efficient design should use the same HW_{SU} irrespective to the number of users in a MU TRX, which can potentially be achieved using HW resource sharing techniques. Hardware Virtualization (HV) is a type of HW resource sharing technique whereby multiple logical instances are created from a single physical instance implemented on HW using time division multiplexing. HV however requires a context switching mechanism that is responsible for storing contexts or states of the current logical instance and restoring the stored contexts or states of the next logical instance upon switching between subsequent logical instances. Thus, HV based HW design for MU TRX need extra HW resources to perform context switching in addition to HW_{SU}. The required HW resources in an MU TRX using the HV approach can be formulated using (3.2),

$$HW_{MU_{HV}} = HW_{SU} + HV_{overhead} \tag{3.2}$$

here, $HW_{MU_{HV}}$ reflects the total amount of HW consumed by the PHY layer of an MU TRX using HV approach, $HV_{overhead}$ represents the extra HW required to perform context switching. The HW_{SU} is constant in (3.2), whereas $HV_{overhead}$ is dependent on N_{users}. Since the value of $HV_{overhead}$ is quite small as compared to HW_{SU}, the overall impact of the increment of $HV_{overhead}$ on $HW_{MU_{HV}}$ is negligible. Hence it is beneficial to use HV in an MU TRX.

In this work, we propose a novel HV based MU TRX design. In order to validate such a design experimentally, $openwifi$ [9]—an existing open-source Wi-Fi implementation on modern SDR platforms—is modified. An SDR is a radio communication system wherein radio components are either running on a general purpose processor (e.g., host computer) or implemented on a programmable hardware (e.g., FPGA), in both cases the key functionalities of radio communication become programmable. Modern SDRs do not only offer flexibility to speed up the validation process, but also support high processing capabilities[1], which is why we selected SDR as the validation platform. The key contributions of this work are summarized below:

1. A complete design of MU transmitter and receiver (or TRX) for OFDMA and MU-MIMO is presented, wherein the HV technique is exploited to keep the HW footprint as close as possible to HW_{SU} irrespective to N_{users}.

[1]The data processing capabilities of the modern FPGA based SDRs are far higher than required by the wireless standards of today. For instance, Zynq Ultrascale+ ZCU102 can provide a maximum Digital Signal Processing (DSP) performance of 5,491 Giga Multiply-Accumulates per Second (GMACS) [10], and a typical HW implementation of IEEE 802.11n standard demands about 23.2 GMACS. Note that 23.2 GMACS is obtained by recompiling the works implemented in [9].

2. A generic methodology that has been followed for applying the HV technique to realise the MU TRX hardware design is described.

3. The applicability of the proposed HW design is experimentally validated for an IEEE 802.11ax compliant MU-OFDMA Transmitter (TX) on the FPGA of a modern SDR, with quantitative analysis in terms of performance and hardware footprint.

The rest of the chapter is organized as follows. Section 3.2 details the state-of-the-art work. The proposed design for the MU TRX is elaborated in Section 3.3. Experimental validation is performed in Section 3.4. Lastly, conclusions and future work are discussed in Section 3.5.

3.2 State of the ART

This section presents the state-of-the-art of radio TRXs, focusing on the realization of MIMO and OFDMA, where the duplication approach is most commonly employed.

3.2.1 SU/MU-MIMO OFDM Transceiver

An IEEE 802.11n compliant 4×4 SU-MIMO OFDM TRX is implemented in Synopsis SAED90nm ASIC [11]. Although the SU-MIMO OFDM TRX serves a SU at a time, it is capable of achieving a throughput of 600 Mbps by transmitting 4 parallel streams in the spatial domain to a SU. The implementation uses 4 dedicated (De)Interleavers, (I)FFT and Guard Interval (GI) modules for each spatial stream, increasing the overall chip area to 13 mm^2. The authors in [11] indeed admit that more optimized approaches are needed to reduce the chip area. Another IEEE 802.11a based 4×4 SU-MIMO OFDM TRX is prototyped on an ASIC in [12]. The authors also addressed the silicon complexity of SU-MIMO OFDM and mentioned that the chip area is increased by a factor of 6.5 when going from SISO to SU-MIMO design using duplication approach.

M. Wenk et al. [8] modified an IEEE 802.11n compliant 4×4 SU-MIMO OFDM TRX [13] on FPGA and presented the testbed for real time MU-MIMO OFDM experiments. In addition to the PHY layer, a polling based MU-Media Access Control (MAC) layer is also added. The testbed provides good insights on how to realize real time MU bidirectional communication, however it uses a dedicated PHY layer chain for each user. Although the paper does not expose quantitative analysis for HW utilization, it is obvious from (3.1) that the HW resources increase proportionally with the number of users.

3.2.2 MU-OFDMA Transceiver

An virtex5 FPGA based Group Orthogonal (GO) OFDMA prototype for high speed variable bit rate broadcast services is presented in [14]. GO-OFDMA is a variant of OFDMA, optimised to reduce the Peak to Average Power Ratio (PAPR). In GO-OFDMA, a subset of OFDM SCs are assigned to different users. In addition, users are grouped into classes depending on the individual data rate requirements. These classes are spread using an orthogonal Walsh Hadamard code. The FPGA based prototype of the GO-OFDMA in [14] consists of 32 points (I)FFT, of which the first 16 SCs are assigned to one user, and the remaining 16 SCs are equally divided between two other users. The corresponding data stream of each user is modulated, spread using an orthogonal code, and interleaved before reaching to Inverse Fast Fourier Transform (IFFT). The processing module before the IFFT module is referred to as a HW chain, which is dedicated for each user, resulting a significant increase in HW resource usage as the number of users rises. Carrier Interferometry (CI) GO-OFDMA [15] is an improved version of GO-OFDMA, providing better performance in terms of PAPR and Packet Error Rate (PER). The virtex5 FPGA based CI-GO-OFDMA prototype is again using the HD approach. Similarly, OFDMA feature of IEEE 802.11ax is implemented on Intel Arria 10 FPGA platform in [16]. However, the HW architecture of OFDMA implementation is not mentioned.

3.2.3 Summary

In the state-of-the-art work concerning parallel data transfer of Wi-Fi, the SU/MU-MIMO or MU-OFDMA are either validated using simulation models or prototyped on an FPGA or ASIC using an HD approach. The simulation based work [17–20] helps to grasp the important concepts of an MU TRX, but only provides insights in non-real-time performance and does not discuss the hardware complexity needed to implement such solutions in a real-life environment.

We observe that existing HW implementations of a MU TRX heavily rely on the HD approach. As shown by (3.1), the HW utilization increases proportionally with the number of users in an MU TRX. Consequently, the MU TRX realised by a HD approach is less economical, as it leads to larger chip area in the final product. Ideally, a MU TRX should not consume more HW resources than a SU TRX. This is the ultimate HW utilization efficiency goal when designing an MU TRX. In this paper, we improve the HW utilization efficiency in an MU TRX by HV. The motivation is to keep the HW utilization of a MU TRX as close as possible to the hardware footprint of a SU TRX. This paper will show that with the proposed MU TRX design, HW_{SU} remains unchanged, while the $HV_{overhead}$ only varies slightly with the number of users. It is proven in subsequent sections that the dependence of $HW_{overhead}$ on N_{user} is insignificant, and therefore can

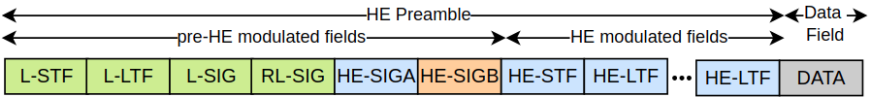


Figure 3.1: The general structure of HE PPDUs in 802.11ax, where HE-SIGB is only present for MU PPDU.

be regarded as a constant. To the best of our knowledge, the use of a MU TRX design using the HV approach that only marginally increases overhead compared to a SU TRX design has not been reported previously in the literature.

3.3 The Proposed Hardware Design for Multiuser Transceiver

In this section, first, the PHY layer Protocol Data Unit (PPDU) formats and the OFDMA feature of IEEE 802.11ax are briefly discussed. Note that the discussion is confined to a 20 MHz Bandwidth (BW) without loss of generality. Next, the procedure for realizing the HV based HW design for the PHY layer of a MU TRX is described. Finally, to validate the applicability of the proposed HW design, an IEEE 802.11ax compliant MU-OFDMA TX is implemented on a modern SDR platform using the proposed HV approach. Its performance and hardware footprint are compared quantitatively against the conventional HD implementation.

3.3.1 802.11ax PPDU Formats

IEEE 802.11ax supports four types of PPDU formats: HE SU PPDU, HE Extended Range (ER) SU PPDU, HE MU PPDU, and HE Triggered Based (TB) PPDU. HE SU PPDU and ER SU PPDU are used for UL/DL transmission to a single user, while HE ER SU PPDU is intended for outdoor scenarios. The remaining two types of PPDU are targeted for MU communication, among which HE MU PPDU is used in DL MU transmission and HE TB PPDU is used in UL MU transmission. Fig. 3.1 shows the generic structure of all the four PPDUs. Each HE PPDU consists of an HE preamble and an HE data part. The HE preamble is comprised of pre-HE modulated and HE modulated fields. The pre-HE modulated field is further composed of Legacy Short Training Field (L-STF), Legacy Long Training Field (L-LTF), Legacy Signal (L-SIG), Repeated Legacy Signal (RL-SIG), HE Signal A (HE-SIGA) and HE Signal B (HE-SIGB). The HE modulated field further consists of HE Short Training Field (HE-STF), HE Long Training Field (HE-LTF). All of these sub-fields in HE preamble part serve specific purposes defined in the IEEE 802.11ax standard. Note that the HE-SIGB sub-field only exists in HE MU PPDU format. For the HE modulated preamble

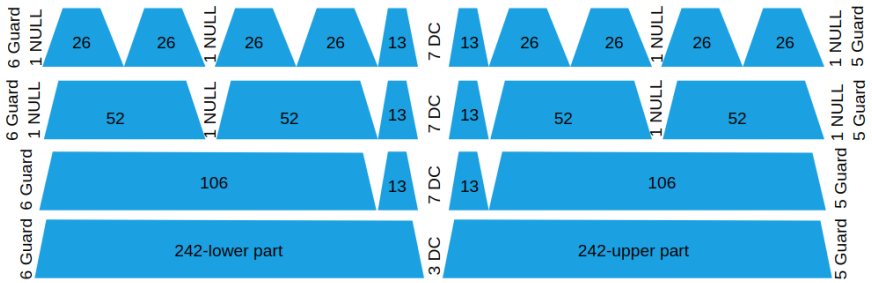


Figure 3.2: Resource units allocation in 20MHz.

and data fields, IEEE 802.11ax uses a 256-point Fast Fourier Transform (FFT) to generate an OFDM symbol with a symbol duration of 12.8 μs and GI options of 0.8 μs, 1.6 μs and 3.2 μs. The pre-HE modulated fields are modulated using a 64-point FFT to form an OFDM symbol with a symbol duration of 3.2 μs and GI of 0.8 μs.

3.3.2 OFDMA in 802.11ax

OFDMA is one of the key technologies introduced in IEEE 802.11ax. It is an OFDM-based multi-access technique, which allocates subsets of SCs (or tones) to different users, allowing simultaneous MU data transmission in the same OFDM symbol. Each group of these SCs is referred to as a RU. IEEE 802.11ax supports 26-tone RU, 52-tone RU, 106-tone RU and 242-tone RU in 20 MHz channel for both UL and DL MU transmission. There are three types of SCs in each RU, namely pilot, data, and unused SCs. While data SCs carries actual data, pilot SCs are known modulated symbols used to perform channel estimation and synchronization at the receiver side. The unused SCs, which can be null, DC, or guard band SCs, do not contain any information. The sole purpose of the unused SCs is to avoid interference from adjacent channels or to reduce the leakage from adjacent RUs within the channel. Fig. 3.2 shows the RU allocation and locations of SCs in each RU within the 20 MHz channel. For instance, IEEE 802.11ax can simultaneously serve up to 9 users by using all 26-tone RUs in the 20 MHz channel. OFDMA makes use of HE MU PPDU and HE TB PPDU for DL MU and UL MU transmissions, respectively. Note that all the RUs in a MU transmission have the same time allocation; i.e., the transmission in each RU ends at the same time. In a DL MU transmission, this is achieved by adding padding bits to the shorter packets. In UL MU transmission, the AP informs the STAs the packet length to be transmitted via the trigger frame.

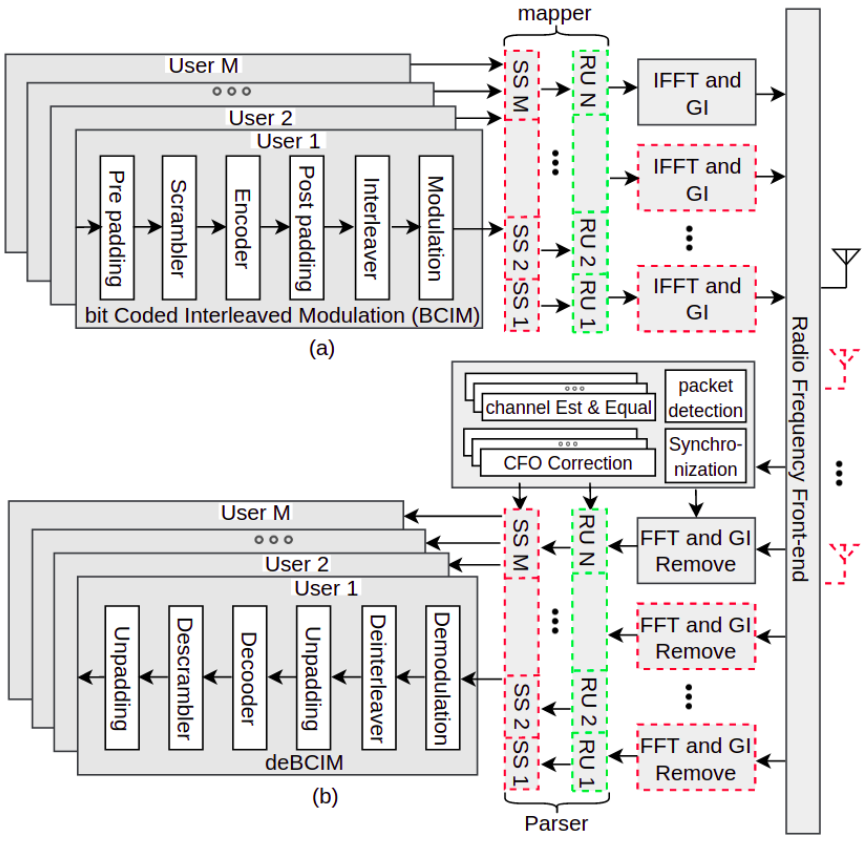


Figure 3.3: The block diagram of the hardware architecture of a MU-OFDMA transceiver (excluding red dotted modules), MU-MIMO transceiver (excluding green dotted modules) and MU-OFDMA MIMO transceiver for IEEE 802.11ax standard. Here, SS, RU and GI denote spatial stream, resource unit and guard interval respectively. For simplicity, a single Spatial Stream (SS) is shown per user, whereas multiple SSs can be assigned to a single RU.

3.3.3 Proposed Hardware Design for the MU Transceiver

The process to realize the proposed HW efficient design for a PHY layer of an MU TRX primarily consists of three steps:

- Drawing the modular design of an MU TRX.
- Identifying modules for which the HW footprint increases with the number of users in the MU TRX, or modules requiring duplication.
- Applying HW efficient techniques (i.e. HV) on the identified modules to keep the HW footprint of MU TRX as close as possible to SU TRX.

Fig. 3.3 shows the block diagram of the HW design of the PHY layer of MU TRX for IEEE 802.11ax standard. The Fig. 3.3-a (excluding modules in red dotted line) illustrates the modules constituting an MU-OFDMA TX. The first module on the TX side, Bit Coded and Interleaved Modulation (BCIM), performs padding (optionally), scrambling, encoding, interleaving and modulation on the data received from higher layers for each user. Next, the frequency mapping module maps users' data to different RUs, which are subsequently combined to form an OFDM symbol in the frequency domain. Thereafter, IFFT is performed to convert it into Time Domain (TD) signal. Lastly, GI is appended to each OFDM symbol, the symbols are sequentially transmitted over the air via the Radio Frequency (RF) front-end. It is apparent that the BCIM module is a critical module because the number of BCIM modules increases with the number of users in the MU TX.

Thus, HW efficient techniques should be applied on the BCIM module. Ideally, a single BCIM module should be used irrespective to the number of users, which is possible using HV technique. HV allows the multiple BCIM logical instances to serve multiple users at the same by using only a single physical BCIM module. Since the proposed HV based design is generic, it can also be applied for MU-MIMO. Fig. 3.3-a (excluding modules in green dotted line) illustrates the modules constituting the PHY layer of an MU-MIMO TX. Note that a single SS is shown per user in Fig. 3.3-a, for simplicity. In addition to the BCIM module, IFFT and GI modules also require HV in the PHY layer of the MU-MIMO TX. This is because for MU-MIMO, each user requires a different signal in the TD, whereas for MU-OFDMA a common TD signal is shared by all users.

The same HV technique is employed in the MU-OFDMA or MU-MIMO Receiver (RX), as shown in Fig. 3.3-b. The majority of the MU RX modules is similar to MU TX, but now we have the reverse process of BCIM module (i.e. deBCIM). Similar to BCIM in MU TX, the deBCIM in MU RX increases with the number of users, hence HV technique is applied on the deBCIM module. Additionally, the MU RX requires synchronization, Carrier Frequency Offset (CFO) correction, channel estimation and channel equalization modules, which are necessary to mitigate the adverse effects on the received signal due to wireless channel impairments and mismatch in carrier frequencies at the TX and RX. The synchronization module serves to find the starting point of a packet, it is shared by all users, whereas CFO, channel estimation and equalization modules increase with the number of users in an MU RX. Therefore, HV technique should also be applied on these modules.

Till this point, we present the complete TRX design for both MU-OFDMA and MU-MIMO, and identified the modules that can benefit from HV. As the procedure to apply HV is generic, it suffices to validate the applicability of the HV based design on one part of the TRX. In the subsequent section, we therefore focus on an OFDMA based MU TX implementation: first, for benchmarking purposes,

the MU TX is designed using the conventional HD approach; next, the design is improved using the HV technique; Finally, the two design approaches, HD and HV, are rigorously compared in terms of HW utilization.

3.3.4 The design for MU-OFDMA TX using Hardware Duplication Approach

Fig. 3.4-a illustrates the complete protocol stack and functional blocks of IEEE 802.11ax PHY for MU-OFDMA TX. The MU-OFDMA TX makes use of the communication stack and a modified version of openwifi's driver to verify the functionality of the proposed PHY layer design. openwifi relies on the existing wireless communication stack (i.e., application, transport, networking and upper MAC layers) of the Linux Operating System (OS) and offers lower MAC layer, radio driver, and IEEE 802.11a/g/n compliant PHY layer. Given that the current Linux kernel running on openwifi has no support for IEEE 802.11ax, we have chosen the packet injection mode, which injects the MAC Payload Data Unit (MPDU) packets directly into the radio driver via mac80211 subsystem and bypass the rest of communication stack in the Linux OS. The radio driver sends the packets to the PHY layer in FPGA via the existing interfaces in openwifi. The PPDU packets are then formed by the PHY layer, the type of the PPDU (i.e., HE SU, MU, TB PPDU) is configured through the driver. Since HE ER PPDU is a variant of SU PPDU, we choose to leave it out of this work for simplicity, however the proposed design can easily support this PPDU format. Finally, the PHY layer is responsible for packet transmission and acknowledging the radio driver about the result of each transmission via interrupts.

The steps needed to generate a PPDU and transmit it through the RF front-end in the PHY layer are summarized as follows and also illustrated in Fig. 3.4:

1. The HE preamble Finite State Machine (FSM) module reads data from the data Block Random Access Memory (BRAM) bank. The data BRAM bank has N numbers of BRAMs with each BRAM dedicated for storing data of a single user. The size of each BRAM is 1024×64 bits, i.e., each BRAM has 1024 locations with each location has a data width of 64 bits. The first few locations of each BRAM contains the configuration needed to form the HE preamble part including the supported coding type, RU size, HE Modulation Coding Scheme (MCS) value, packet length for each RU, GI value, HE-LTF types. Based on these configurations, the FSM formulates L-SIG, RL-SIG, HE-SIGA, HE-SIGB, HE-STF, and HE-LTF fields.

2. Subsequently, BCIM module performs scrambling, encoding, interleaving, and modulation for the PPDU headers. Note that only the BCIM module of the first user is utilized to generate the PPDU header, as it is common

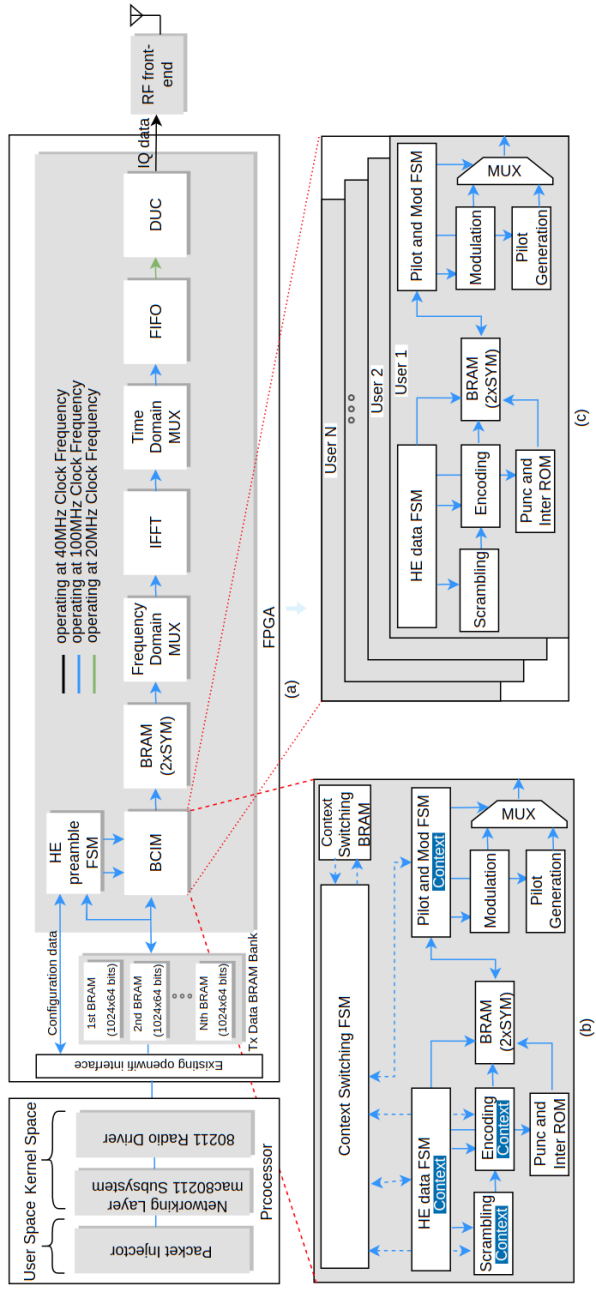


Figure 3.4: The proposed design for (a) MU-OFDMA transmitter using (b) hardware virtualization and (c) hardware duplication approaches.

for all users (see User 1 BCIM module in Fig. 3.4-c). In addition, HE-STF, HE-LTF fields skip the BCIM modules, as their modulated version in the frequency domain is stored in the Look Up Tables (LUTs) of FPGA. Till this point, the modulated SCs of the HE preamble part is generated, which is common for all users.

3. In order to generate HE data part, the HE data FSM in the BCIM module reads the user specific data from the data BRAM bank. Then, the same scrambling, encoding, interleaving, and modulation steps are performed on the user specific data. Since packets' size can be different for each user and MU transmission of all users end at the same time, HE data FSM is also responsible for padding if a user's packet cannot occupy all the OFDM symbols. Unlike PPDU header generation, a dedicated BCIM module is used to generate the user specific HE data and number of BCIM modules used depend on the number of users in a MU transmission (see Fig. 3.4-c).

4. Next, Frequency Domain Multiplexer (FD-MUX) combines the modulated SCs (i.e., either from the common HE preamble part or from HE data part of different users), pilot SCs, and unused SCs and generates a 64-point or 256-point OFDM symbol in the frequency domain, depending on whether or not the symbol belongs to the pre-HE modulated fields.

5. Finally, the IFFT module generates OFDM symbols in the TD. The Time Domain Multiplexer (TD-MUX) appends a GI for each OFDM symbol, and the Digital Up Converter (DUC) upsamples and transmits the signal over the air via the RF front-end. In our design, TD samples of L-STF and L-LTF fields of the pre-HE preamble part are stored in LUTs of the FPGA. The TD-MUX reads these LUTs first, and then uses samples generated by IFFT to form a complete HE PPDU packet.

6. Upon transmission of all the OFDM symbols, the PHY layer generates an interrupt for the radio driver and waits for the next packet.

All the modules in the PHY layer are pipelined. A high level illustration of the pipelining is shown in Fig. 3.5, wherein the horizontal and vertical axis represent the time scale and modules of the PHY layer, respectively. At $t0$ time, HE preamble and BCIM modules begin processing the first OFDM symbol. After a tick, these modules start processing the second OFDM symbol and FD-MUX begins processing the first OFDM symbol. The tick is defined as a time unit required to process the Number of coded bits per OFDM symbol ($OFDM_{CBPS}$), which is equal to $\sum_{u=1}^{N_{users}} N_{CBPS}(u)$. In other words, N_{users} BCIM physical modules (in case of HD approach) or logical modules (in-case of HV approach) are required to generate a single $OFDM_{CBPS}$. The value of $N_{CBPS}(u)$ depends on the MCS

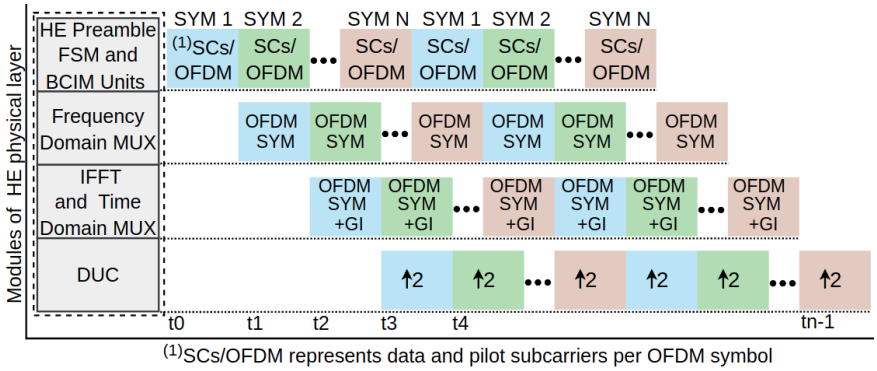


Figure 3.5: Applying pipelining at module level.

value and RU size for each user. After the third tick (i.e., $t3$), all the modules are busy in processing the consecutive OFDM symbols. The BRAMs (one between the interleaver and modulation submodules of BCIM module, and the other between the BCIM and FD-MUX) have the capacity to store 2 OFDM symbols and behaves as a ping pong buffer. That is, the BRAMs can accept a new OFDM symbol while sending out the previous OFDM symbol. The pipeline based design does not only improve the latency by parallel running the modules, but it also allows to increase the speed of system clock at which the baseband processing is performed, which further increases the processing throughput.

3.3.5 The Design for MU-OFDMA TX using Hardware Virtualization Approach

The HV approach based MU-OFDMA TX uses the same communication stack and functional blocks shown in Fig. 3.4-a except for the BCIM module. Unlike the HD approach where a dedicated BCIM module is utilized for each user, the HV approach utilizes the same BCIM module for multiple users. This is achieved by using multitasking, pipelining and multi-clock domains. This subsection addresses only the BCIM module, as the remaining modules do not need to be duplicated, hence are already fully discussed in the previous subsection.

3.3.5.1 Multitasking

Multitasking commonly used in the field of OS enables OS to execute multiple tasks at a time by rapid context switching among tasks. During each context switching, the contexts or states of the current task are saved and the states of the next task are restored, requiring extra HW or software resources to perform save-restore operations. We aim to apply the multitasking concept to utilize the

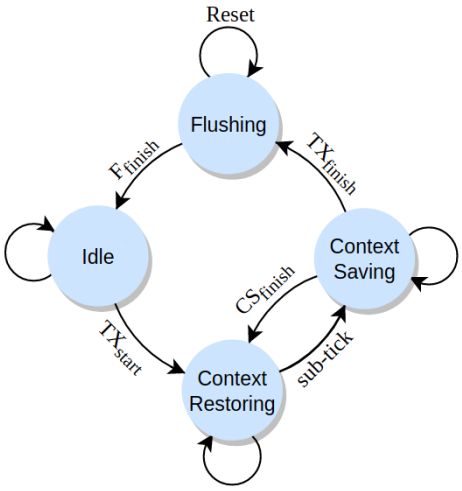


Figure 3.6: State diagram of finite state machine used for context switching.

same BCIM module for multiple users. However, it also inherits the context save-restore problem. Like a dedicated program and memory unit used in OS to perform context save-restore operation, a dedicated Context Switching Finite State Machine (CS-FSM) and an extra Context Switching Block Random Access Memory (CS-BRAM) is introduced in the proposed design (see Fig. 3.4-b). The first step to design the CS-FSM is to identify the submodules in the BCIM module which need context save-restore operations. In principle, any submodule which has memory, delay, internal states or pipelines demands a context switching. Four modules, which are highlighted with $context$ in Fig. 3.4-b, in the proposed design need to save the context for the current user and restore the context for the next user during each context switching operation.

The CS-FSM is responsible for the correct functioning of all these operations. The state diagram for CS-FSM is drawn in Fig. 3.6. At the beginning of a packet transmission, the CS-FSM enters into $Flushing$ state in which CS-BRAM is initialized with the reset state of the context switching-enabled submodules in the BCIM module. After $N_{users} \times 4$ clocks, it enters into $Idle$ state and waits for the TX_{start} signal generated by low MAC. The CS-FSM allocates 4 memory locations of CS-BRAM for each user, with each location being 64-bits wide. These locations are used to store the internal states of HE data FSM, data scrambler, convolutional encoder, and the FSM used to insert pilots and perform modulation; in addition to the internal states in the BCIM chain, this memory is also used to store user-specific configurations such as RU size, MCS value, packet length per user, pre-(or post-)padding length per user. When the TX_{start} is issued, the CS-FSM instructs the BCIM to begin the data processing and then enters the

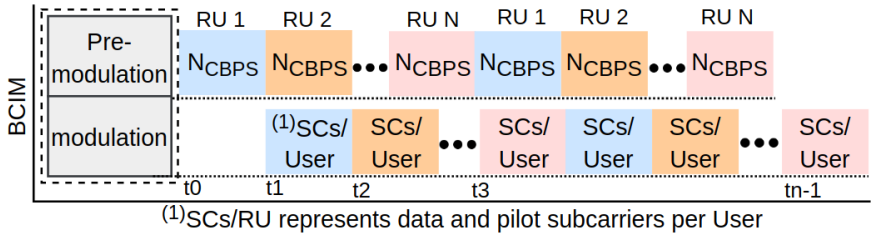


Figure 3.7: Applying pipelining at sub-module level.

$ContextRestoring$ state. After a sub-tick, CS-FSM restores the states of the next user and enters into the $ContextSaving$ state. The sub-tick is defined as a time required to process $N_{CBPS}(u)$. In the $ContextSaving$ state, the CS-FSM stores the states of the current user to be used later. After four clock cycles, the CS-FSM generates CS_{finish} signal indicating that the context switching is finished and re-enters into $ContextRestoring$ state. The CS-FSM also checks whether an iteration has been performed for all RUs in an OFDM symbol. If this condition is met, it increments the generated number of OFDM symbols, which is common for all users. Once all the OFDM symbols are generated, the CS-FSM generates TX_{finish} signal and goes back to the $Flushing$ state.

3.3.5.2 Pipelining

The HV approach leverages pipelining at two different levels. One pipeline operates at the OFDM symbol rate, which is shown in Fig. 3.5 and explained in the previous subsection, the other pipeline is shown in Fig. 3.7. The second pipeline works inside the BCIM module and is managed by the CS-FSM. The submodules of the BCIM module are categorized into pre-modulation, and modulation with each having a different processing load in a single context switching operation. The pre-modulation part consists of HE data FSM, scrambling, encoding, and interleaving submodules; whereas modulation, pilot generation, and FSM responsible for pilot insertion and modulation are the parts of the modulation category (see Fig. 3.4-c). In each context switching operation, the pre-modulation part needs to process $N_{CBPS}(u)$ of an user, while modulation needs to modulate all SCs (i.e., data and pilot SCs per user) in the assigned RU. Thus, a BRAM is used to decouple these two parts and it has a capacity to store two $OFDM_{CBPS}$ data. The value of $N_{CBPS}(u)$ depends on the MCS and RU size of a given user, whereas the value of SCs/RU relies only on the RU size. At t0 time, all the submodules of pre-modulation part begin the data processing of the first user. After a sub-tick (i.e., time equivalent to process $N_{CBPS}(u)$), the pre-modulation part switches to the second user, meanwhile the submodules of the modulation part are enabled for the first user. The BRAM acts as a ping pong buffer, i.e., it accepts the output of

pre-modulation while providing input to the modulation submodules. At the same moment, pre-modulation and modulation submodules are working concurrently, but for different users.

3.3.5.3 Multi-clock Domains

The design make uses of three different clocks to meet the timing constraints required by MU-OFDMA TX (see Fig. 3.4-a). All the modules up to TD-MUX operate at 100 MHz, whereas DUC reads data at 20 MHz and generates IQ samples at 40 MHz. A First-in First-out (FIFO) is placed between DUC and TD-MUX to mitigate metastability. Metastability is a phenomenon which happens when data crosses multiple clock domains and causes the output to be unpredictable.

It is worth noting that, independent on the approach (HD or HV) the BCIM module of the MU-OFDMA TX operates at the same 100 MHz clock speed. It does not comply to the fundamental theory behind the virtualization concept, which demands overclocking in order to generate multiple virtual instances from a single physical instance. In other words, the BCIM module should theoretically operate at $N_{users} \times 100$ MHz. For instance, based on this theory the operating frequency of BCIM should be 900 MHz to simultaneously serve 9 users with each mapped to a 26-tone RU in 20 MHz BW, however in reality the BCIM module still operates at 100 MHz. This can be explained as follows, given the fixed total BW in MU-OFDMA TX, the data rate of each RU decreases, as the number of RUs rises. For the BCIM module, the amount of clocks required for processing depends on a user's MCS value. The higher the MCS, the more information bits are coded into a symbol. This work leverages the original openwifi's modulation module, which only supports up to MCS 6. In the worst case scenario, 9 26-tone RUs in 20 MHz channel with each RU having MCS 6 requires ($N_{RU} \times N_{CBPS}(u) = 9 \times 144$) 1296 clocks to generate a single OFDM symbol in the frequency domain, while a single 242-tone RU in a 20 MHz channel with MCS 6 demands ($N_{RU} \times N_{CBPS}(u) = 1 \times 1404$) 1404 clocks. Thus, the number of clocks to process multi users' data could be less than a single user. The in-depth understanding of the OFDMA operation allows us to design a MU-OFDMA TX using HV without overclocking.

3.4 Results and Discussions

In this section, the Proof of Concept (PoC) developed to validate the designs for a MU-OFDMA TX using HV is discussed. First, the supported features of the MU-OFDMA TX are summarized. Next, the performance of the HV based MU-OFDMA TX is benchmarked against the HD based MU-OFDMA TX in terms of

Table 3.1: Supported features for MU-OFDMA transmitter

PPDU type	x-tone RU size	MCS values	Max RUs
HE SU PPDU	242	0-6	1
HE TB PPDU	242, 106, 52, 26	0-6	1
HE MU PPDU	242, 106, 52	0-6	4

HW utilization. Finally, the experimental results of the PoC validation are presented.

3.4.1 The Supported Features in the PoC

The proposed design for MU-OFDMA TX is validated for a 20 MHz channel. The supported features of the PoC are listed in Table 3.1 and are applicable for both HD and HV approaches. Three types of HE PPDUs are supported, namely the HE SU PPDU, HE TB PPDU, and HE MU PPDU, which are used for UL/DL SU transmission, UL MU transmission and DL MU transmission, respectively. It is worth noting that the PoC does not support 26-tone RU in HE MU PPDU, meaning that DL MU transmission is limited to 4 simultaneous users. On the other hand, UL MU transmission is possible for all RU sizes defined in IEEE 802.11ax within a 20MHz channel. Finally, as the original openwifi can perform modulation up to MCS 6, this limitation also applies to our PoC.

3.4.2 Comparison of Hardware Utilization

The proposed design is validated on Zynq UltraScale+MPSoC ZCU102 Evaluation Kit [21]. The Zynq MPSoC on ZCU102 evaluation kit is composed of Programmable Logic (PL) (or FPGA) and Processor System (PS) (or ARM Cortex-A53). The PHY layer of the MU-OFDMA TX is implemented on FPGA. The implemented design on FPGA consists of three parts:

- logical parts, which maps to LUTs and Registers (which can be either Latches or Flip-Flops (FFs)) in FPGA,
- memory part, which is placed on dedicated BRAMs in the FPGA,
- signal processing part (e.g., multiplication, division, etc), which maps on dedicated Digital Signal Processing (DSP) blocks in FPGA.

Thus, the hardware utilization efficiency between the HD approach and HV approach is calculated in terms LUTs, Registers, BRAMs and DSPs, based on the FPGA compilation reports for both approaches. The obtained results are shown in Table 3.2. The efficiencies (i.e., the improvement in hardware resource utilization

Table 3.2: Hardware utilization comparison between a MU-OFDMA transmitter designed using a hardware duplication approach and a MU-OFDMA transmitter designed using a hardware virtualization approach.

N_{RU}	Resource	HW_{HV}	HW_{HD}	$\eta_{\%}$	η_n	$HV_{overhead}$
1	LUTs	59528	49351	-20.6	0.83	10177
	Registers	15931	15626	-1.95	0.98	305
	BRAMs	11.5	10.5	-9.52	0.91	1
	DSPs	21	21	0.00	0.00	0
2	LUTs	59566	92361	35.51	1.55	10215
	Registers	15998	24253	34.04	1.52	372
	BRAMs	11.5	18.5	37.84	1.61	1
	DSPs	21	25	16.00	1.19	0
3	LUTs	59583	135394	55.99	2.27	10232
	Registers	16066	32885	51.15	2.05	440
	BRAMs	11.5	26.5	56.60	2.30	1
	DSPs	21	29	27.59	1.38	0
4	LUTs	59667	178441	66.56	2.99	10316
	Registers	16136	41517	61.13	2.57	510
	BRAMs	11.5	34.5	66.67	3.00	1
	DSPs	21	33	36.36	1.57	0

of HV approach relative to HD approach) shown in Table 3.2 are calculated using (3.3) and (3.4).

$$Improved\ Efficiency(\eta_{\%}) = \left(\frac{HW_{HD} - HW_{HV}}{HW_{HD}}\right) \times 100 \tag{3.3}$$

$$Normalized\ Efficiency(\eta_n) = \left(\frac{HW_{HD}}{HW_{HV}}\right) \tag{3.4}$$

where, HW_{HD} and HW_{HV} represent the HW consumed by HD and HV approaches, respectively. The HW can be LUTs, Registers, BRAMs or DSPs. The ZCU102 evaluation kit has 912 BRAMs, each has 36 Kbs capacity in the FPGA, and the number of BRAMs shown in the Table 3.2 reflects the number of 36Kbs BRAMs used. Similarly, there are 2520 DSP48E2 blocks in the evaluation kit[2]. The number of DSPs in the Table 3.2 represents the number of used DSP48E2 blocks.

It can be seen from Table 3.2 that the efficiency of the proposed HV approach becomes more visible as the number of users (or RUs) increases. Note that the efficiency of the HV approach is worse than HD approach for a single user. This is however in line with our expectations, because the proposed approach involves

[2]A DSP48E2 block consists of 27 bits pre-adder, 27 $\times$18 bits signed multiplier and 48 bits datapath multiplexer.

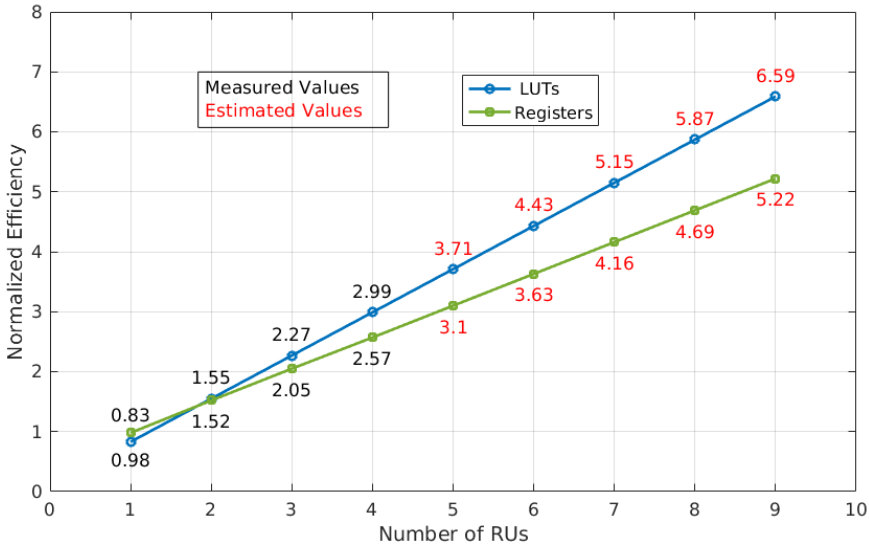


Figure 3.8: The normalized efficiency of HV based MU-OFDMA TX, estimated for up to 9 users (i.e., the maximum amount of users in 20 MHz bandwidth.

extra logic and memory for context switching. However, most part of the HW used to implement context switching remains constant as the number of users increase in MU-OFDMA TX, resulting higher HW utilization efficiency for higher numbers of users. Thus, the best efficiency is achieved when maximum numbers of users are present, among the supported scenarios this is the case of 4 users, where the HV based MU-OFDMA TX design consumes ≈3 times less LUTs and 2.57 times less Registers as compared to HD based MU-OFDMA TX. Similarly, the HD based MU-OFDMA TX uses 3 times more BRAMs and 1.57 times more DSPs than the HV based MU-OFDMA TX.

Though the proposed HV based MU-OFDMA TX is validated for up to 4 users, the HW consumption for higher number of users can be estimated by using (3.2). The last column of Table 3.2 contains the HW overhead due to HV (i.e., $HV_{overhead}$). The $HV_{overhead}$ mainly reflects to the HW consumed by the CS-FSM and CS-BRAM submodules. These submodules do not require any DSP operation, the consumed DSPs for $HV_{overhead}$ remains zero irrespective to the N_{users}(or N_{RU}). Thus, $HV_{overhead}$ due to DSPs is not added in the estimation results. In addition, Fig. 3.8 does not include η_n in terms of BRAMs as well. The CS-FSM in the the proposed HV based MU-OFDMA TX uses four 64 bits locations for each user for context save-restore operations. The OFDMA of IEEE 802.11ax can support up to 9 users in 20 MHz channel. At most 2304 bits (64 $\times$ 9 $\times$ 4) memory is needed in the estimation for the case of 9 users, which is

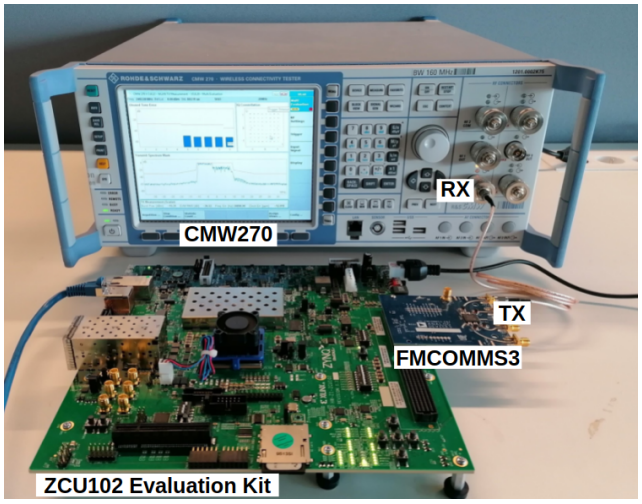


Figure 3.9: The experimental setup used for the real-life validation of MU-OFDMA transmitter.

less than the capacity of a single BRAM (i.e. 36 Kbs) of the used FPGA. Thus, $HV_{overhead}$ due to BRAM in the estimation up to 9 users is always 1, hence not shown in Fig. 3.8.

Using the known values of $HV_{overhead}$ and HW_{SU} for up to 4 users, the HW normalized efficiency (η_n) can be approximated via (3.2) and (3.4). The estimated η_n in terms of LUTs and Register for up to 9 users is shown in Fig. 3.8. The η_n for the consumed LUTs and Registers almost follows a straight line (i.e., $y = m \times x + c$) with different slopes. Fig. 3.8 shows that the HD based MU-OFDMA TX costs ≈ 6.5 times more LUTs and ≈ 5.2 times more Registers than the HV based MU-OFDMA TX, in case of 9 users in a 20 MHz channel.

3.4.3 Experimental Validation

The experimental setup used to validate the PoC is shown in Fig. 3.9. The setup consists of an SDR board where the HV based MU-OFDMA TX is running. A Rohde & Schwarz CMW270 [22] wireless connectivity tester is used to verify the functionality of the MU-OFDMA TX in a real-life environment. The SDR is composed of the ZCU102 evaluation kit and FMCOMMS3 [23], an analog RF front-end. While the PHY layer of the MU-OFDMA TX is implemented on the PL part of the evaluation kit, upper layers including radio driver are running on the PS part of the kit. The existing communication stack of openwifi is leveraged and modified to verify the functionality of the proposed HV based MU-OFDMA TX. The MU-OFDMA TX is configured in packet injection mode [24], which allows

Table 3.3: The EVM performance of MU-OFDMA transmitter.

MCS	HE SU/MU PPDU		HE TB PPDU	
	EVM_S(dB)	EVM_M(dB)	EVM_S(dB)	EVM_M(dB)
0	-5	-38.92	-27	-40.26
1	-10	-38.83	-27	-40.36
2	-13	-38.69	-27	-41.05
3	-16	-38.8	-27	-41.43
4	-19	-38.86	-27	-41.02
5	-22	-38.98	-27	-40.87
6	-25	-38.31	-27	-40.69

to send IEEE 802.11ax compliant packets over the air. On the other side, the CMW270 tester, which is configured in non-signaling IEEE 802.11ax compliant mode, decodes the packets transmitted by the SDR and measures the performance of the MU-OFDMA TX in terms of Error Vector Magnitude (EVM), unused tones error, and spectral mask.

3.4.3.1 Error Vector Magnitude

EVM is used to quantify the performance of MU-OFDMA TX and is a measure of how accurately it transmits the symbols within its constellation diagram. The requirements defined in IEEE 802.11ax to measure the EVM performance is the same for HE SU PPDU and HE MU PPDU. The EVM is measured by receiving a minimum of 20 PPDUs with each at least 16 symbols long if the RU size is greater than the 26-tone RU (which is the case for our PoC which does not support 26-tone RU in HE MU PPDU). Table 3.3's left part shows the measured EVM (EVM_m) values against the EVM values (EVM_s) defined in the standard for HE SU/MU PPDU. The measured EVM_m values of MU-OFDMA TX transmitting SU PPDU and MU PPDUs are obtained using the Rohde & Schwarz CMW270 wireless tester. In addition, all the possible combinations of 52-tone, 106-tone and 242-tone RUs are transmitted using the RU Allocation vector defined in the IEEE 802.11ax standard (see Table 27-26 in [4] for more details) and measured. In general the EVM performance shows rather consistent results under different MCS and RU configurations, the worst level observed is -38.17 dB. A typical measurement is shown in Table 3.3, obtained when MU-OFDMA TX is transmitting an MU PPDU with a RU Allocation vector equal to 112; i.e., 4 52-tone RU, each RU assigned with MCS 6 (the highest supported MCS value in the PoC). It is evident from the Table 3.3 that the HV based MU-OFDMA TX outperforms the EVM requirements defined in the standard.

Unlike MU/SU PPDU, The EVM in HE TB PPDU is measured by receiving 20 PPDUs with each at least 32 symbols long, as the PoC supports now a 26-tones

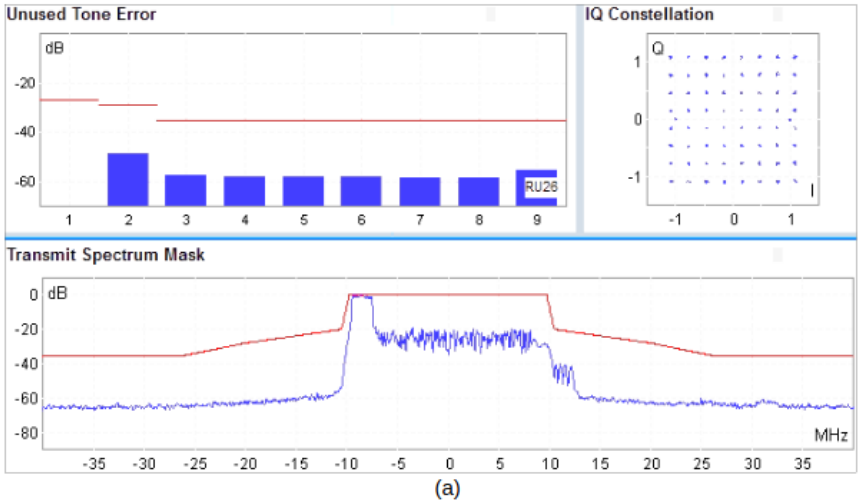


Figure 3.10: Measured performance of the MU-OFDMA TX in terms of EVM, unused tone error, and spectral mask using a Rohde & Schwarz CMW270 tester when the MU-OFDMA TX implemented on the SDR is transmitting (a) HE TB PPDU, which contains a single 26-tone RU (RU1) and (b) HE MU PPDU, which contains 4 52-tone RUs.

RU in a HE TB PPDU. The EVM performance in TB PPDU is measured for both occupied RUs and unoccupied RUs. The EVM performance of unoccupied RUs is termed as relative EVM (or unused tone error), which is explained in the subsequent subsection. The last two columns of Table 3.3 depicts the EVM values of the occupied RUs for HE TB PPDU. The worst EVM_m value for TB PPDU is -40.03 dB. Again we observe very stable EVM performance under different RU

configurations and MCS settings. The EVM_m values shown in Table 3.3 for TB PPDU are measured when the MU-OFDMA TX is transmitting a TB PPDU on the 26-tone RU1 using MCS 6. The index "1" of the RU indicates the location of the 26-tone RU in the 20 MHz channel. Such kind of information (i.e. RU index, channel width) are sent by an AP to a STA using the trigger frame before the actual UL transmission happens.

3.4.3.2 Unused Tones Error

The relative EVM limit (or unused tones error) is a new performance metric introduced in IEEE 802.11ax standard to quantify the performance of unoccupied RUs in TB PPDU. The main objective behind introducing the unused tones error is to minimize the interference between adjacent RUs. An example of HE TB PPDU is shown in Fig. 3.10-a, where data is being transmitted on the 26-tone RU1 using MCS 6. That is, the first 26-tone RU is the occupied RU and the rest of the 26-tone RUs are unoccupied 26-tone RUs. The $Unused\ Tone\ Error$ graph in Fig. 3.10-a visually explains the the EVM of occupied 26-tone RU (i.e., tone error) and relative EVM of unoccupied 26-tone RUs (i.e., unused tones error) in the form of staircase mask (the red line shows what is defined in the standard, and the blue bars represents the staircase mask of the MU-OFDMA TX). The unused tones error is the EVM for an unoccupied 26-tone RU. The IEEE 802.11ax standard specifies the values of unused tones errors with respect to the EVM (or tone error) of the occupied RU(s). The unused tones errors follow the relative staircase mask defined in IEEE 802.11ax standard (see Equation 27-131 in [4] for more details) in units of 26-tone RU to avoid frequency dependent variations. Note that the $Unused\ Tone\ Error$ graph for HE MU PPDU in Fig. 3.10-b is empty, because the graph is only valid for HE TB PPDU.

3.4.3.3 Spectral Mask

The spectral mask helps to reduce the adjacent-channel interference by defining the limits of out-of-band emission. It is provided in the units of Decibel relative (DBr) (i.e., dB relative to maximum spectral density of the signal). Fig. 3.10 shows the spectral masks for HE MU PPDU (see Fig. 3.10-b) and HE TB PPDU (Fig. 3.10-a). The HE MU PPDU in Fig. 3.10-b is transmitting on 4 52-tone RUs with each 52-tone RU carries different users' data. A dip in Fig. 3.10-b corresponds to the location of the central 26-tone RU, which is not occupied. Similarly, the HE TB PPDU in Fig. 3.10-a is transmitting data on 26-tone RU1 and the remaining eight 26-tone (i.e., RU2 to RU9) are unused. It is evident from the spectral mask of Fig. 3.10 that the MU-OFDMA TX meets the spectral mask requirements defined in IEEE 802.11ax standard.

3.5 Conclusions

This paper introduces a novel hardware efficient design for UL/DL MU transceiver. Unlike the traditional duplication approach where a dedicated physical layer is allocated for each user, the proposed design creates multiple logical physical layers based on a single physical layer by leveraging the hardware virtualization technique. The design is highly efficient in terms of hardware utilization, hence could effectively reduce the die area of a final commercial chip.

To validate the proposed design, an MU-OFDMA transmitter is implemented on an SDR platform. The MU-OFDMA transmitter is compliant to the IEEE 802.11ax standard and supports HE SU PPDU, HE MU PPDU and TB PPDU. The performance of the MU-OFDMA transmitter has been quantified in terms of spectral mask, EVM and unused tones by using a Rohde & Schwarz CMW 270 tester configured in the IEEE 802.11ax mode. The experimental results unveil that the MU-OFDMA transmitter outperforms the requirement specified in the IEEE 802.11ax standard, at the same time it saves LUTs by a factor of $\approx$3 and Registers by a factor of 2.57 hardware resources for the specific case of 4 RUs wherein each RU is transmitting data for different users.

The proposed design is validated with the IEEE 802.11ax compliant MU-OFDMA transmitter, however it has a potential to be applied for any hardware modules requiring duplication. It can be applied to any transceiver or standard involving MU communication such as IEEE 802.16, LTE, and 5G. In the future, we are planning to complete the implementation of IEEE 802.11ax compliant MU-OFDMA receiver and also toward the MU-MIMO, resulting a “total duplication-free” 802.11ax chip design.

Acknowledgment

This research was funded by the Flemish FWO SBO S003921N VERI-END.com (Verifiable and elastic end-to-end communication infrastructures for private professional environments) project and the Flemish Government under the ”Onderzoeksprogramma Artificiële Intelligentie (AI) Vlaanderen” program.

References

[1] K. O. M. Salih, T. A. Rashid, D. Radovanovic, and N. Bacanin. *A Comprehensive Survey on the Internet of Things with the Industrial Marketplace*. Sensors, 22(3), January 2022. Available from: ttps://www.mdpi.com/1424-8220/22/3/730, doi:https://doi.org/10.3390/s22030730.

[2] *Wi-Fi Alliance® 2022 Wi-Fi® trends*, 2021. Available from: https://www.wi-fi.org/news-events/newsroom/wi-fi-alliance-2022-wi-fi-trends.

[3] *802.11-2020 - IEEE Standard for Information Technology–Telecommunications and Information Exchange between Systems - Local and Metropolitan Area Networks-Specific Requirements - Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications.*, 2020.

[4] *802.11ax-2021 - IEEE Standard for Information Technology–Telecommunications and Information Exchange between Systems Local and Metropolitan Area Networks–Specific Requirements Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications Amendment 1: Enhancements for High-Efficiency WLAN*, 2021.

[5] *802.16-2017 - IEEE Standard for Air Interface for Broadband Wireless Access Systems*, 2017. Available from: https://ieeexplore.ieee.org/document/8303870.

[6] *The Mobile Broadband Standard*, 2022. Available from: https://www.3gpp.org/.

[7] K. Nishimori, R. Kudo, N. Honma, Y. Takatori, and M. Mizoguchi. *16x16 multiuser MIMO testbed employing simple adaptive modulation scheme*. In VTC Spring 2009-IEEE 69th Vehicular Technology Conference, pages 1–5, Spain, 2009. IEEE. Available from: https://ieeexplore.ieee.org/document/5073280, doi:10.1109/VETECS.2009.5073280.

[8] M. Wenk, P. Luethi, T. Koch, P. Maechler, N. Felber, W. Fichtner, and M. Lerjen. *Hardware platform and implementation of a real-time multi-user MIMO-OFDM testbed*. In 2009 IEEE International Symposium on Circuits and Systems, pages 789–792, Taiwan, 2009. IEEE. Available from: https://ieeexplore.ieee.org/document/5117872, doi:10.1109/ISCAS.2009.5117872.

[9] J. Xianjun, L. Wei, and M. Michael. *open-source IEEE802.11/Wi-Fi baseband chip/FPGA design*, 2021. Available from: https://github.com/open-sdr/openwifi.

[10] *UltraScale Architecture and Product Data Sheet: Overview*, 2021. Available from: https://www.xilinx.com/support/documentation/data_sheets/ds890-ultrascale-overview.pdf.

[11] T. H. Tran, Y. Nagao, M. Kurosaki, B. Sai, and H. Ochi. *ASIC design of 600Mbps 4×4 MIMO wireless LAN system*. In 2012 14th International Conference on Advanced Communication Technology (ICACT), pages 360–363, South Korea, 2012. IEEE.

[12] D. Perels, S. Haene, P. Luethi, A. Burg, N. Felber, W. Fichtner, and H. Bolcskei. *ASIC implementation of a MIMO-OFDM transceiver for 192 Mbps WLANs*. In Proceedings of the 31st European Solid-State Circuits Conference, 2005. ESSCIRC 2005., pages 215–218, France, 2005. IEEE. Available from: https://ieeexplore.ieee.org/document/1541598, doi:10.1109/ESSCIR.2005.1541598.

[13] S. Haene, D. Perels, and A. Burg. *A Real-Time 4-Stream MIMO-OFDM Transceiver: System Design, FPGA Implementation, and Characterization*. IEEE Journal on Selected Areas in Communications, 26(6):877–889, 2008. doi:https://doi.org/10.1109/JSAC.2008.080805.

[14] A. Agarwal, V. K. Sinha, R. Palisetty, P. Kumar, K. C. Ray, K. Kumar, and T. Pandey. *Performance analysis and FPGA prototype of variable rate GO-OFDMA baseband transmission scheme*. Wireless Personal Communications, 108(2):785–809, 2019. doi:https://doi.org/10.1007/s11277-019-06429-4.

[15] V. Mukati, A. R. Khan, B. V. Reddy, and P. Kumar. *A variable rate multi-user CI/GO-OFDMA scheme over frequency selective channel*. In 2014 IEEE International Conference on Advanced Networks and Telecommuncations Systems (ANTS), pages 1–5, India, 2014. IEEE. Available from: https://ieeexplore.ieee.org/document/7057260, doi:10.1109/ANTS.2014.7057260.

[16] S. Kim, M. M. Rashid, S. Deo, J. Perez-Ramirez, M. Galeev, G. Venkatesan, S. Dey, W. Li, and D. Cavalcanti. *Demo/poster abstract: Enabling time-critical applications over next-generation 802.11 networks*. In IEEE INFOCOM 2018 - IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS), pages 1–2, USA, 2018. IEEE. Available from: https://ieeexplore.ieee.org/document/8406889., doi:10.1109/INFCOMW.2018.8406889.

[17] Y. Daldoul, D.-E. Meddour, and A. Ksentini. *Performance evaluation of OFDMA and MU-MIMO in 802.11ax networks*. Computer Networks, 182:107477, 2020.

[18] D. Magrin, S. Avallone, S. Roy, and M. Zorzi. *Validation of the ns-3 802.11 ax OFDMA Implementation*. In Proceedings of the Workshop on ns-3, pages 1–8. ACM, 2021.

[19] O. Seijo, J. A. Lopez-Fernandez, and I. Val. *w-SHARP: Implementation of a High-Performance Wireless Time-Sensitive Network for Low Latency and Ultra-low Cycle Time Industrial Applications*. IEEE Transactions on Industrial Informatics, 17(5):3651–3662, 2021.

[20] *802.11ax Waveform Generation*, 2021. Available from: https://nl.mathworks.com/help/wlan/ug/802-11ax-waveform-generation.html.

[21] *Zynq UltraScale+ MPSoC ZCU102 Evaluation Kit*, 2021. Available from: https://www.xilinx.com/products/boards-and-kits/ek-u1-zcu102-g.html.

[22] *CMW270 wireless connectivity tester*, 2021. Available from: https://www.rohde-schwarz.com/us/products/test-and-measurement/wireless-tester-network-emulator/rs-cmw270-wireless-connectivity-tester_63493-9552.html.

[23] *AD-FMCOMMS3-EBZ: An AD9361 Software Defined Radio Board*, 2021. Available from: https://www.analog.com/en/design-center/evaluation-hardware-and-software/evaluation-boards-kits/eval-ad-fmcomms3-ebz.html.

[24] *How to use packet injection with mac80211*, 2021. Available from: https://www.kernel.org/doc/html/latest/networking/mac80211-injection.html.

4

An Approach to Achieve Zero Turnaround Time in TDD Operation on SDR Front-End

Chapter 2 & 3 focus on Hardware (HW) efficient designs for Multi User (MU) Transceivers (TRXs) using modern Software Defined Radios (SDRs). However, the slow Turnaround Time (TT) of the analog RF front-ends of modern SDRs make them inadequate when used to design the high-performance wireless communication standards operating in Time Division Duplex (TDD) mode. Thus, in this chapter, a novel solution is presented to reduce the TT on an SDR.

⋆ ⋆ ⋆

Muhammad Aslam, Xianjun Jiao, Wei Liu, and Ingrid Moerman.

Published in IEEE Access, Vol. 6, pp. 75461-75470, Nov. 2018.

Abstract Thanks to the digitization and softwarization of radio communication, the development cycle of new radio technologies can be significantly accelerated by prototyping on Software Defined Radio (SDR) platforms. However, a slow Turnaround Time (TT) of the front-end of an SDR for switching from receiving mode to transmitting mode or vice versa, is jeopardizing the prototyping of

wireless protocols, standards, or systems with stringent latency requirements. In this chapter, a novel solution called BaseBand processing unit operating in Half-Duplex mode and analog Radio Frequency front-end operating in Full-Duplex mode, BBHD-RFFD, is presented to reduce the TT on SDR. A prototype is realized on the widely adopted AD9361 radio frequency front-end to prove the validity of the proposed solution. Experiments unveil that for any type of application, the TT in Time Division Duplex (TDD) operation mode can be reduced to zero by the BBHD-RFFD approach, with negligible impact on the communication system in terms of receiver sensitivity. The impact is measured for an in-house Institute of Electrical and Electronics Engineers (IEEE) 802.15.4 compliant transceiver. When compared against the conventional TDD approach, only a 7.5-dB degradation is observed with the BBHD-RFFD approach. The measured sensitivity of -91 dBm is still well above the minimum level (i.e., -85 dBm at 2.4 GHz) defined by the IEEE 802.15.4 standard.

4.1 Introduction

Software Defined Radio (SDR) is increasingly used in diverse wireless applications, thanks to its flexible and programmable features compared with traditional radio chips. From the day of its invention, it has been successfully adopted for experimental performance evaluation within the wireless research community. An SDR is primarily composed of 2 parts: a programmable digital component and a configurable analog Radio Frequency (RF) front-end. The digital component, which consists of a number of digital Baseband Processing Units (BBPUs) implemented on programmable devices, such as Digital Signal Processor (DSP), a Field Programmable Gate Arrays (FPGA), or a Central Processing Unit (CPU). The reconfigurable analog RF front-end of an SDR is simply everything between the BBPUs and the antenna. The general components pertaining to an RF front-end are amplifier, mixer, filters, Digital-to-Analog Converter (DAC), and Analog-to-Digital Converter (ADC).

When implementing a wireless standard that uses Time Division Duplex (TDD), the SDR needs to be switched regularly between Receiving (Rx) and Transmitting (Tx) modes. The switching time or Turnaround Time (TT) is defined as the time required by the Physical (PHY) layer to change from Rx mode to Tx mode or vice versa. During TT, the components in the analog RF front-end are powered up and stabilized, which consumes a considerable amount of time. This time consumption becomes more critical for low latency feedback applications, such as process control loops in industrial systems where remote control of robotic arms or other machineries are involved. The influence of TT on latency of a wireless system in a simple Device-to-Device (D2D) topology is illustrated in Fig. 4.1. It is evident that during TT there is no data transfer, and the wireless channel capacity

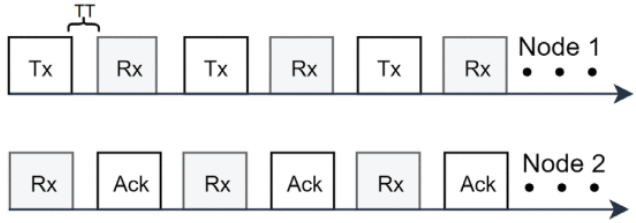


Figure 4.1: The impact of turnaround time on latency during TDD operation in a Device-to-Device (D2D) scenario.

is wasted. Normally, the negative impact of TT becomes more severe when smaller packets and faster switching scenarios are considered. For example, a special technique proposed in [1] actually leverages the fast and frequent Tx/Rx switching to achieve "virtual Full-Duplex". In this solution, a Tx or Rx slot is in the order of microseconds, and only a few physical layer symbols instead of a complete packet is transferred in a slot. Since the Tx/Rx switching occurs very frequently, it appears as if both nodes are transmitting and receiving concurrently, creating the perception of real Full-Duplex (FD) realized with a half-duplex implementation. This solution is however based on the assumption that TT is in the order of nanoseconds, and is only validated in simulation environment. Unfortunately TT of an SDR front-end is generally much higher (i.e., $\approx$18 μs in case of AD9361 [2]). AD9361 introduced by Analog Devices is an RF transceiver chip widely used on SDR RF front-ends, including several devices (e.g., B200 and E310) in the popular Universal Software Radio Peripheral (USRP) family [3], and a range of the FMCOMMS boards [2, 4, 5]. As such, a short TT cannot be taken for granted on the current mainstream SDR platforms. This is one of the main differences between SDR and commercial transceiver chipsets today.

The TT not only plays an important role for a single D2D link scenario, but also for low latency operation in larger scale networks involving many wireless nodes sharing the same spectrum. Several common wireless standards implementing TDD clearly have to cope with hardware limitations related to TT. For instance, the Short Interframe Spacing (SIFS) in the Wireless Fidelity (Wi-Fi) standard, which is the maximum interval a transceiver will wait for receiving the first symbol of the acknowledgement after sending a packet, is limited by the duration of the TT [6]. The SIFS of IEEE 802.11a/g is 10 μs , whereas the 60 GHz Wi-Fi standard (IEEE 802.11ad) has a shorter SIFS of 3 μs. As mentioned earlier, the switching time of an SDR's RF Front-End (SDR RF-FE) is much higher, which makes it infeasible for implementing standards requiring a stringent TT at Media Access Control (MAC) level.

In summary, (i) squeezing the TT directly reduces the latency of a D2D wireless link, (ii) a sufficiently short TT is a prerequisite to realize wireless standards

with stringent timing requirements, and (iii) the TT of today's SDR RF-FE is significantly larger than the majority of commercial wireless chipsets and therefore reduction of the TT on an SDR RF-FE is essential for prototyping existing state-of-the-art and future new wireless standards.

In this chapter, a novel approach called BBHD-RFFD – BaseBand processing unit operating in Half-Duplex mode and analog Radio Frequency front-end operating in Full-Duplex mode – is introduced to reduce the TT on an SDR RF-FE. The remainder of this chapter is organized as follows: Related works and motivation are discussed in Section 4.2, implementation of the BBHD-RFFD approach is presented in Section 4.3, performance comparison with traditional TDD approach is made in Sections 4.4 and 4.5, and concluding remarks are given in Section 4.6.

4.2 Related Work and Motivation

Many high-end wireless standards have stringent requirements in terms of latency to ensure the correctness of MAC protocols (some MACs even require precision of response time at the granularity of microseconds. For example, SIFS value of Wi-Fi standards). Commercially available radio chips that are built on DSPs or Application Specific Integrated Chips (ASICs) easily satisfy these requirements as they are optimized for these specific wireless protocols and specific spectral bands. Most SDRs, on the other hand, are generic and are optimized to operate in a wide range of spectral bands and for a wide range of wireless technologies. Despite the overall decent performance of SDRs, they are generally less optimized in comparison to DSPs or ASICs. For example a typical ASIC for Wi-Fi has a TT of 1 μs [7], whereas it is 18 μs for AD9361. While 18 μs is considerably low, it is not sufficient to support Wi-Fi or self-contained TDD operation in 5$^{\text{th}}$ Generation (5G) New Radio (NR) μs [8].

SDRs can be, in general, categorized into two groups: General Purpose Processor (GPP) based SDR, and non-GPP based SDR. The following sections discuss latency performance of the two groups.

4.2.1 GPP Based SDR

GPP based SDR usually consists of an RF front-end, ADC/DAC, embedded up and down converters on FPGA or DSP boards, and a GPP based host machine. A bridge interface, such as Ethernet or Universal Serial Bus (USB), is used to exchange the radio samples between GPP and the radio. In these SDRs, most of the baseband signal operations are offloaded to GPP. USRP N-Series are the examples of such architectures [3]. USRPs are modular so they can deal with the applications operating from DC up to 6 GHz. While their performance is suitable for research experiments and quick prototyping, these platforms do not necessarily

meet the requirements of time critical communication standards. For instance, the authors of [9] measure the latency of a number of USRPs. The measured round trip latency between RF front-end and host computer for N210 is 103 μs . This latency does not include any preprocessing of the data which would have to be included in the latency calculations of a specific protocol implementation. This relatively high latency makes GPP based SDR impracticable for time critical standards and applications (e.g., SIFS value of Wi-Fi).

4.2.2 non-GPP Based SDR

In a non-GPP based SDR architecture, in general, while PHY layer is implemented on hardware (e.g., FPGA or DSP), the MAC layer is implemented on an embedded processor. The USRP Embedded (E) series (USRP E-Series), USRP X-Series [3], Xilinx FPGA with FMCOMMSx RF front-end board (FMCOMMS SDR) [2], and the WARP v3 [10] are the examples of such architecture. Where, USRP E-Series incorporate Xilinx System on Chips (SoCs) to develop standalone SDR, and WARP v3 contains a Xilinx Virtex-6 FPGA, which includes two MicroBlaze processors. The compact architecture of non-GPP based SDR makes it more suitable for prototyping any low latency wireless standard.

The non-GPP based SDRs can be sub-categorized into two groups: narrowband non-GPP based SDRs, wideband non-GPP based SDRs.

4.2.2.1 Narrowband Non-GPP Based SDR

Narrowband non-GPP based SDRs are capable of prototyping low latency wireless protocols, but in limited spectrum. The main hindrance in such kind of SDR is its RF front-end. For example, any time critical communication standards realized on WARP SDR platform would be operating only at 2.4 GHz or 5 GHz Industrial, Scientific and Medical (ISM) bands. This is because WARP SDR platform employs MAX2829 chipset [7] as RF front-end, which can only operate on the aforementioned ISM bands. The WARP 802.11 reference design [11] is an example of implementation on WARP platform, which meets the SIFS requirements of this standard.

4.2.2.2 Wideband Non-GPP Based SDR

On the other hand, USRP E-Series, USRP X-Series and FMCOMMS SDR platforms, which are examples of wideband non-GPP based SDR, can be operated over relatively wide spectrum, because they utilize commodity Wideband RF (WB-RF) front-ends. For instance, USRP E310 and Xilinx FPGA with FMCOMMS2 board incorporate AD9361 as a RF front-end that can be operated up to 6 GHz. It is observed that WB-RF front-ends have relatively high TT (e.g., $\approx$18 μs in AD9361),

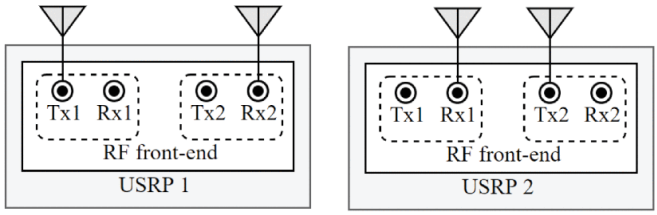


Figure 4.2: Antenna connections in 802.11 application framework.

which makes them unfeasible for prototyping the time-critical wireless standards (for example: SIFS values of Wi-Fi standards). In a nutshell, a wideband SDR platform with negligible TT (fast switching Tx to Rx or vice versa) is always desired. In literature, a very few real-time SDR based works have been presented.

Authors in [12] use Xilinx Zynq FPGA with AD9361 Radio Frequency Front-end (RF-FE) to prototype the IEEE 802.11a standard, but their implementation is not real-time. Wu et al. [13] has exploited the AD9371 SDR RF-FE [14], which is the next generation of AD9361 and it has the same TT as that of AD9361, to implement the Wi-Fi standard. They introduce a new SDR architecture based on hardware and software co-design, called tick programmable low latency SDR system in their work. IEEE 802.11ac SISO and MIMO and full-duplex 802.11a/g are implemented. They claim that the SIFS requirement of 802.11ac standard can be achieved with their proposed implementation by giving extensive measurements both at MAC and PHY layers. However, their break-down analysis does not include TT of AD9371 RF-FE. They prototype a real-time system with separate Rx and Tx setups. Since the Rx and Tx setups do not switch their state, TT of RF-FE does not affect the performance of their setup (which is not a case of real-life system). To conclude, they claim that their implementation for IEEE 802.11ac SISO meets SIFS requirement, without including the TT of RF-FE, which is inevitable for systems operating in our real-life.

Recently, IEEE 802.11 Application Framework [15] introduced by National Instruments (NI) is able to meet SIFS. It, however, uses 2 USRPs with each having two RF transceivers. In this setup, one of the two RF transceivers of an USRP configures in Rx mode only, while other in Tx mode only. As shown in Fig. 4.2, Rx port of one USRP is connected with the Tx port of other and vice versa. This setup does not need to switch RF transceiver between Tx and Rx mode (i.e., zero TT), but it employs more than one transceivers to achieve this. We however achieve the same results with one transceiver. Throughout this chapter we use SDR for wideband non-GPP based SDR.

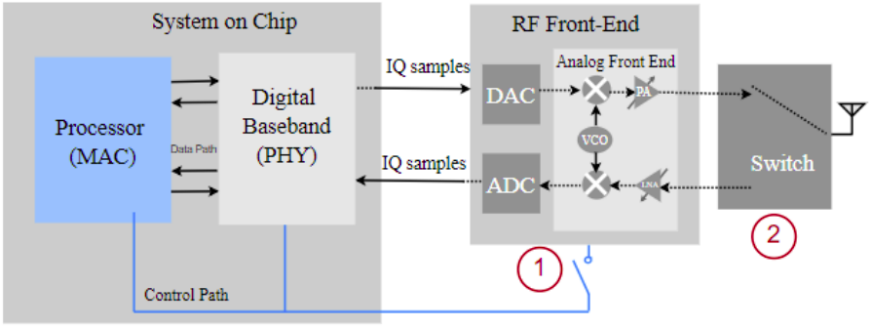
In this chapter, we have performed an extensive study on SDR RF-FE and proposed a unique approach that enables researcher to implement any low latency application on a mainstream SDR platform.

4.3 The Proposed Solution

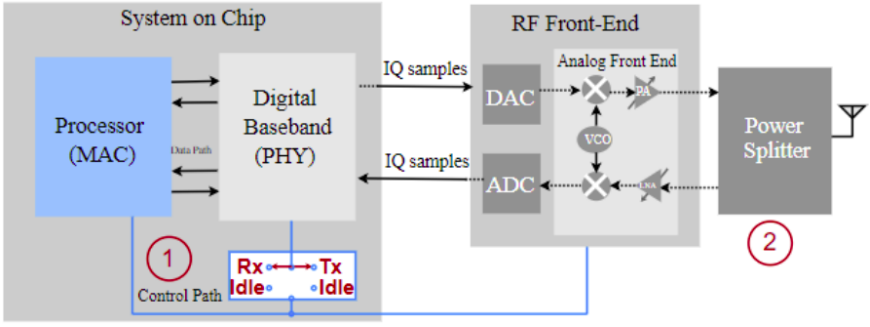
To decrease the TT in TDD operation in SDR devices, this chapter explores a new paradigm called BBHD-RFFD – baseband processing unit operates at half-duplex, while RF front-end at FD mode. The conventional way to configure the RF front-end for TDD operation (referred to as conventional TDD approach hereafter), is presented in Fig. 4.3-a for the ease of comparison and discussion, whereas Fig. 4.3-b illustrates our proposed approach.

Fig. 4.3-a and Fig. 4.3-b both have the basic blocks in the radio communication stack, namely the MAC, PHY and RF front-end. For experimental validation we have implemented MAC and PHY on a "System on Chip" hardware architecture. More information on this implementation is provided in the next paragraph. The exact way how MAC and PHY are realized, is not relevant for the BBHD-RFFD approach, the only thing that matters is the control path (blue lines) towards the "RF Front-End". In a conventional TDD approach, the respective RF front-end components are switched on or off upon transition between Tx and Rx modes, as indicated by ① in Fig. 4.3-a. Since most of the RF components are analog, they consume more time to be turned on and stabilized than digital components. The turn-on time in combination with their software commands considerably contribute to the overall TT of a wireless system.

In the BBHD-RFFD approach, the RF components of transmitter and receiver remain on all the time, thus no time is required to turn on the analog components. Apparently, this configuration comes with more power consumption, therefore we also introduce Power Saving Scheme (PSS) in the BBHD-RFFD approach. More information on the PSS can be found in Section 4.5. In this case, switching only occurs in digital baseband domain, as indicated by ① in Fig. 4.3-b. In general, physical layer functionality is comprised of multiple BBPUs. Essentially, the Tx/Rx switching comes down to activating the respective BBPUs either in the Tx or the Rx path. The software controlling the switching ensures the half-duplex operation by enabling only one set of BBPUs at a time. Since Tx and Rx paths have separate BBPUs, the respective path can be attached to the corresponding In-phase and Quadrature phase (IQ) samples of the radio board. For instance, in Tx mode, BBPUs related to Tx path are attached to IQ samples of DAC, meanwhile BBPUs related to Rx path are detached from IQ samples of ADC. Thanks to microelectronics, the time consumed for digital switching action takes a few nanoseconds. Our digital system is operating at 100MHz, therefore it takes only 10 ns to switch between Tx and Rx modes. The proposed scheme is however not applicable for most single-standard commercial off-the-shelf chips. For instance, the IEEE 802.15.4 compliant chip CC2538 cannot turn on Tx and Rx related RF components at the same time [16], hence it can only support the conventional TDD approach.



(a) Conventional TDD Approach



(b) BBHD-RFFD Approach

Figure 4.3: Block diagram of (a) the conventional TDD and (b) the proposed BBHD-RFFD approaches.

The BBHD-RFFD approach is realized on an off-the-shelf SDR hardware platform as follows. The FMCOMMS2 [2] board with AD9361 chipset is exploited as SDR RF-FE, while IEEE 802.15.4 [17] standard compliant PHY and ALOHA MAC layer are implemented in the Programmable Logic (PL) and Processor System (PS) parts of a Zynq-7000 All Programmable System on Chip (AP SoC) (i.e., ZedBoard in our setting), respectively [18]. The FMCOMMS board is usually accompanied by a Xilinx FPGA board to form a complete SDR platform, which is also the case in our setup. The FMCOMMS2 board has separate Tx and Rx antenna ports, a Mini-Circuits' ZN2PD2-63+ power splitter [19] is used to attach both Tx and Rx ports to the antenna, as shown at ② in Fig. 4.3-b, without using an antenna switch as shown at ② in Fig. 4.3-a. The function of the power splitter is twofold, (i) it allows us to transmit or receive the RF signal via a single antenna; and (ii) it gives an isolation of more than 20 dB between the Tx and Rx ports. This isolation is important to our solution, because when leaving the RF components constantly activated, the Rx path is impacted by self-interference even when the

Tx BBPUs are deactivated. In this chapter, the DC leakage and internal noise of various components in both transmit and receive path are collectively defined as self-interference. The power splitter is a low cost and effective approach to mitigate the impact of self-interference, as is further detailed in Section 4.4.

This chapter also investigates how the different operation modes of the AD9361 chipset can be used for the conventional TDD and the BBHD-RFFD approaches. The AD9361 can be operated in the following three duplex modes:

- TDD WithOut Calibration (TDD-WOC): In this mode, RF components are calibrated once during initialization phase, afterwards the components specific to Tx or Rx chains are simply turned on without calibration. Calibration refers to activities such as tuning the Voltage Controlled Oscillator (VCO) and waiting for the Phase Locked Loop (PLL) to stabilize around the desired output frequency.

- TDD With Calibration (TDD-WC): In this mode all the RF components are turned on and then calibrated every time when switching between Tx and Rx occurs.

- Frequency Division Duplex (FDD): In this mode, all the RF components are turned on and calibrated during initialization in FDD mode, and all the RF components pertaining to Rx and Tx chains remain operational afterwards. Since both Tx and Rx paths remains on all the time, there is no need of switching (Tx to Rx or vice versa) in FDD mode.

In this chapter, both TDD-WC or TDD-WOC can be exploited to implement the conventional TDD approach. It is recommended to repeat calibration in AD9361 each time the operating channel frequency is changed. Thus, TDD-WC should be employed in a situation where regular change of Tx/Rx frequency occurs (e.g., Bluetooth). The FDD mode of AD9361 is exploited to implement the BBHD-RFFD approach.

In addition to the three duplex modes, the 'DC offset calibration and tracking feature' of AD9361 helps to minimize the DC offset in the received IQ samples, while the 'quadrature calibration and tracking' feature aims to maintain orthogonal relation between the received I and Q samples by performing IQ correction [20]. The aforementioned calibration and tracking features of AD9361 are hereinafter referred to as DC/IQ tracking, which is further explored in experiments to suppress the DC component in the self-interference.

4.4 Experiments

In this section, we first measure the amount of self-interference under different settings, and derive the optimal setting to implement the BBHD-RFFD approach,

and then experiments are conducted to benchmark our proposed approach against the conventional TDD approach, in terms of performance in TT and receiver sensitivity.

4.4.1 Analysis of Self-Interference

4.4.1.1 Measurement Setup

Fig. 4.4 shows the setup utilized to measure the self-interference under different conditions and configuration of AD9361, an RF front-end in the FMCOMMS2 board used in our experimental setting. All measurements in this section use 2.41 GHz as center frequency. AD9361 has two identical and independently controlled transmit and receive paths, which are both shown in Fig. 4.4, though only one set of Tx/Rx channel is used in the measurement. The measurement is performed in the following 3 steps:

- unintended RF signals (i.e., self-interference) pass through Receiving (Rx) chain of AD9361. In this chain, the I and Q samples obtained at the output of ADC, operating at 8 Msps, are filtered by a low pass Finite Impulse Response (FIR) filter with cutoff frequency of 2.6 MHz[1],
- Integrated Logic Analyzer (ILA), a logic analyzer core that can be used to monitor the internal signals of a design, in FPGA (the PL part of Zynq 7000 SoC) is used to capture the filtered IQ samples,
- post processing of the collected IQ samples is done in MATLAB to obtain the Power Spectrum Density (PSD) and the average power of the IQ samples, using (4.1) and (4.2) respectively.

$$\begin{aligned} &PSD = 10 \times \log_{10}\left(|Y_k|^2\right) \quad where \\ &Y_k = \sum_{n=0}^{N-1} \left((I_{fir} + iQ_{fir})_n\right) \times e^{-\frac{2\pi i}{N}kn},\ 0 \le k \le N-1 \end{aligned} \tag{4.1}$$

$$P_{avg} = 10 \times \log_{10}\left(\frac{1}{N}\sum_{n=0}^{N-1}\left(I_{fir}[n]^2 + Q_{fir}[n]^2\right)\right) \tag{4.2}$$

where I_{fir} or Q_{fir} denote the filtered IQ samples at the output of AD9361 FIR filter, and N denotes the total number of collected IQ samples. The PSD is obtained by squaring the outputs (i.e., Y_k) of the Fourier Transform. The P_{avg} 's unit is dB, because it has not been calibrated. However this suffices our purpose to compare the strength of self-interference under various conditions.

[1]We choose 2.6 MHz as the cut off frequency because we later on use IEEE 802.15.4 compliant PHY to measure the receiver sensitivity, 2.6 MHz is adequate to offer the signal bandwidth defined in the standard.

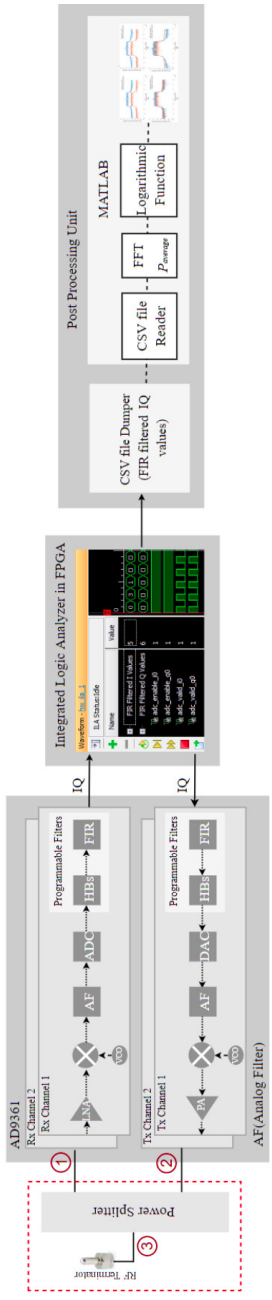


Figure 4.4: Experimental setup to capture IQ samples from AD9361 and post processing.

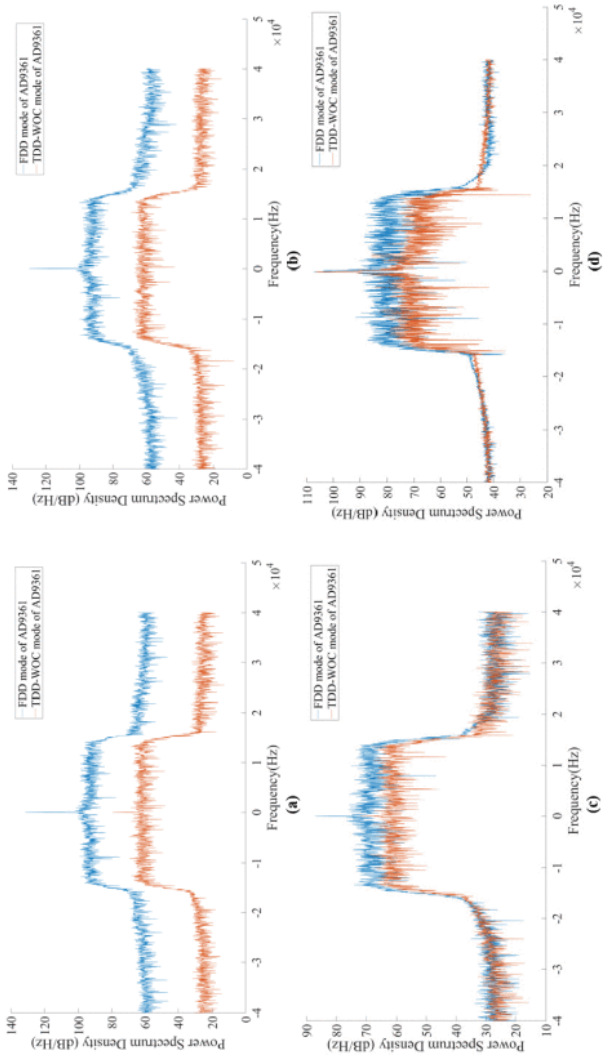


Figure 4.5: The comparison of self-interference (visualized by PSD), between FDD and TDD-WOC modes of AD9361, under the following settings: (a) Tx and Rx ports are directly connected and DC tracking is disabled; (b) Tx and Rx ports are directly connected and DC tracking is enabled; (c) Tx and Rx ports are connected through a power splitter with DC tracking enabled; (d) Tx and Rx ports are connected to two antennas, placed orthogonal to each other.

The self-interference when no transmission takes place has been analysed with PSD measurements for the TDD-WOC and FDD modes of AD9361 for the following 4 cases:

Table 4.1: Self-interference measured by power difference in the 4 Cases.

Case	P_{fdd}(dB)	$P_{tdd-woc}$(dB)	P_{diff}(dB)
Case 1	65.75	26.84	38.9
Case 2	64.43	26.85	37.6
Case 3	34.65	26.45	8.2
Case 4	47.23	44.03	3.2

- **case 1** (see Fig. 4.5-a): DC/IQ tracking is disabled, Tx and Rx ports (① and ② in Fig. 4.4) are directly connected without any power splitter
- **case 2** (see Fig. 4.5-b): DC/IQ tracking is enabled, Tx and Rx ports are directly connected without any power splitter
- **case 3** (see Fig. 4.5-c): DC tracking is enabled, Tx and Rx ports are connected via power splitter, the 3rd port of power splitter is terminated, to prevent external interference, as shown in the dashed red rectangular of Fig. 4.4
- **case 4** (see Fig. 4.5-d): DC tracking is enabled, Tx and Rx ports (① and ② in Fig. 4.4) are attached to two antennas, positioned orthogonal to each other. Extra measures have been taken to ensure that the operational band is clean at the time of the experiment.

As there is no self-interference in the TDD-WOC mode, the average power difference between FDD and TDD-WOC modes represents the intensity of the self-interference. This is calculated according to (4.3)

$$P_{diff} = P_{fdd} - P_{tdd-woc} \tag{4.3}$$

where P_{fdd} and $P_{tdd-woc}$, denote P_{avg} obtained during FDD and TDD-WOC modes, respectively. The P_{diff} for all the aforementioned four cases are listed in Table 4.1.

4.4.1.2 Measurement Analysis

Among the 4 cases, **case 1** is used as a benchmark, as it does not involve any effort to remove self-interference. When comparing **case 2** with **case 1**, the self-interference of FDD mode is reduced by 1.3 dB, thanks to the DC tracking feature of AD9361. When comparing **case 3** with **case 1**, the self-interference of FDD mode is suppressed by 30.7 dB. The high-quality isolation (i.e., observed 29.4 dB at 2.41 GHz) between two ports of the power splitter, together with the DC tracking feature in **case 3** helps to mitigate the effect of self-interference of FDD mode. Although the power splitter has strongly reduced the self-interference, it also causes the Tx power degradation by a factor of 3.6 dB, due to the insertion

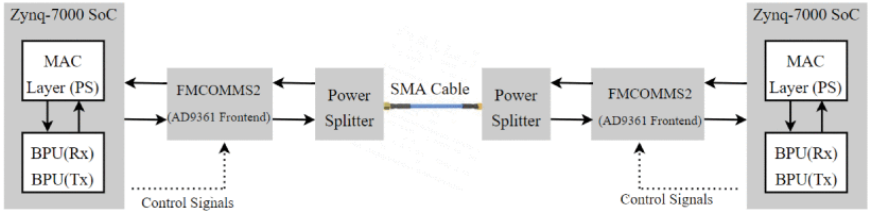


Figure 4.6: Experimental setting for turnaround time and receiver sensitivity measurements.

loss of the splitter. The observed self-interference reduction in **case 4** is further improved by 5 dB compared to case 3, indicating that this setup offers the better results in terms of performance. Case 4 has the advantage of less Tx power loss, however it requires accurate antenna placement (orthogonal) to achieve optimal performance, whereas **case 3** is easier to implement and likely to offer more repeatable measurements.

In conclusion, the setup in **case 3** – TDD-WOC and FDD modes of AD9361 along with power splitter and DC tracking enabled feature – has been selected to implement the conventional TDD approach and BBHD-RFFD approach, respectively.

4.5 Analysis of Turnaround Time and Receiver Sensitivity

4.5.1 General Measurement Setup

The experiment setup used for TT and receiver sensitivity measurements is illustrated in Fig. 4.6, consisting of two identical SDR nodes. In terms of hardware, each SDR node is composed of a Zynq-7000 SoC (i.e, ZedBoard in our setting), an FMCOMMS2 board which incorporates AD9361 front-end, and a power splitter. A SubMiniature version A (SMA) cable is used to connect the two SDR nodes, ensuring interference-free condition for the sensitivity measurement. Each SDR node is configured to act as an IEEE 802.15.4 compliant transceiver.

4.5.2 TT Measurement and Analysis

The TT measurement is relatively simple; it requires only one SDR node, TT is determined by accessing relevant registers of AD9361. More specifically, the following steps are used to measure the TT:

- A MAC application running on PS (as shown in Fig. 4.6) issues commands to AD9361 to trigger Tx/Rx switching,

```
--Rx To Tx Switching  -- Tx To Rx Switching
1: GetTime(t0)        1: GetTime(t0)
2: ENSM(TX)           2: ENSM(RX)
3: GetTime(t1)        3: GetTime(t1)
4: CalFunc(t1-t0)     4: CalFunc(t1-t0)
```

Figure 4.7: Pseudo codes for the TT measurements.

- in parallel a global timer on PS is enabled and started,
- the MAC application continuously read the status register of AD9361 (i.e., polling),
- upon obtaining the desired status (i.e., polling ended, and related registers of Rx or Tx have changed their status), the global timer is stopped and TT is calculated by reading the current value of the global timer.

Fig. 4.7 summarizes the aforementioned procedure used for TT measurements with pseudo code. *GetTime()* is used to get the current values of the running global timer, while ENable State Machine (ENSM) controls the switching state of AD9361. *CalFunc()* provides the calculated TT consumed during Tx to Rx switching or vice versa. The ENSM command switches the AD9361 to the desired state specified in its input parameter. There are two possible states of ENSM in our implementation: with RX as input parameter, the AD9361 is switched into Rx mode, while the input parameter TX forces the AD9361 into Tx mode. Table 4.2 shows the TT consumed by all three modes of AD9361 whenever switching takes place. It can be observed that Rx to Tx switching always takes more time because the turn-on time of DAC is higher than that of ADC in AD9361. The worst TT values are observed in TDD-WC mode whereas nearly zero (i.e., $\approx$10 ns) TT is observed in FDD mode (i.e., BBHD-RFFD approach). The maximum TT specified by the IEEE 802.15.4 standard is 192 μs [17], therefore this standard can be implemented with any of the three modes of AD9361. The SIFS values for IEEE 802.11 standard are 10 μs (2.4GHz) [6] and 3 μs (60GHz) [6]. Both TDD-WC and TDD WOC of AD9361 would fail to meet SIFS (TT is one of the many parts of SIFS) requirements of Wi-Fi, but our proposed approach makes this feasible. In fact, any standard with stringent TT can be realized with our approach.

4.5.3 Receiver Sensitivity Measurement and Analysis

As mentioned before, since RF parts of both transmitter and receiver are constantly activated in our approach, the receiving path is suffering from self-interference. In section 4.4 the self-interference has been analyzed to be 8.2 dB and this causes

Table 4.2: The average TT measured under three duplex operation modes of AD9361 with a standard deviation of Around 1 μs.

TT (μs)	TDD-WC	TDD-WOC (Conventional TDD)	FDD (BBHD-RFFD)
Tx to Rx	40.2	4.2	$\approx$0
Rx to Tx	53.0	17.2	$\approx$0

the noise to increase by the same amount. It is therefore expected that the receiver sensitivity of the system will be equally affected. We are using IEEE 802.15.4 compliant BBPUs, therefore the sensitivity is measured according to this standard requirements [17]. The sensitivity of a receiver is defined as the input power level when the Packet Error Rate (PER) drops to 1%. PER is measured by counting the proportion of packets lost during transmission against 10,000 transmitted packets, each containing 20 octets with 14 bytes PHY payload. In order to determine the receiver sensitivity, the PER measurement is repeated when packets are sent with different Tx powers, which is controlled by tuning the variable attenuator (i.e., PA block in Fig. 4.3-b) in the AD9361 Tx chain.

For determining receiver sensitivity, however, PER needs to be mapped to the input power of the receiver rather than Tx attenuation. The translation from the Tx attenuation (dB) to absolute Rx power (dBm) is achieved using the following steps: (1) the SMA cable along with power splitter is detached from the receiver AD9361 RF front-end and attached to an Anritsu MS2690 A spectrum analyzer [21], in another word, the spectrum analyzer replaces the receiver SDR in the setup depicted in Fig. 4.6, (2) the Rx power is measured via spectrum analyzer when different settings of the attenuator are applied to AD9361. Fig. 4.8 shows that the relation between the Tx attenuation and the measured Rx power by the spectrum analyzer is approximately linear. By making use of this linear relation, Tx attenuation can be easily translated into absolute power level, which is then used to acquire the receiver sensitivity.

The PER obtained under a range of Rx input power are illustrated in Fig. 4.9. It can be observed from Fig. 4.9 that the receiver sensitivity of BBHD-RFFD and the conventional TDD are -91 dBm and -98.5 dBm, respectively. The measured receiver sensitivity of the conventional TDD approach is comparable with that of commercial chipsets. For instance, cc2538 chipset has the receiver sensitivity of -97 dBm [16]). It can be concluded from the measurements that the receiver sensitivity of our approach is degraded by 7.5 dB, compared to the conventional TDD. This degradation is obviously caused by the 8.2 dB self-interference, however it is not exactly aligned (i.e., 0.7 dB difference). This is because DC leakage is a part of the self-interference, which does not affect the receiver sensitivity. In conclusion, despite the degradation, the receiver sensitivity in BBHD-RFFD approach is still

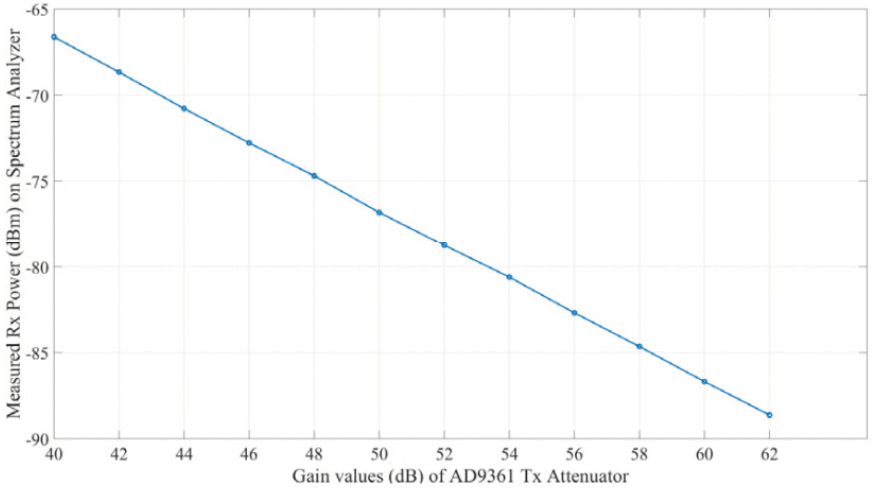


Figure 4.8: Measured Rx power (in dBm) vs AD9361 Tx attenuator.

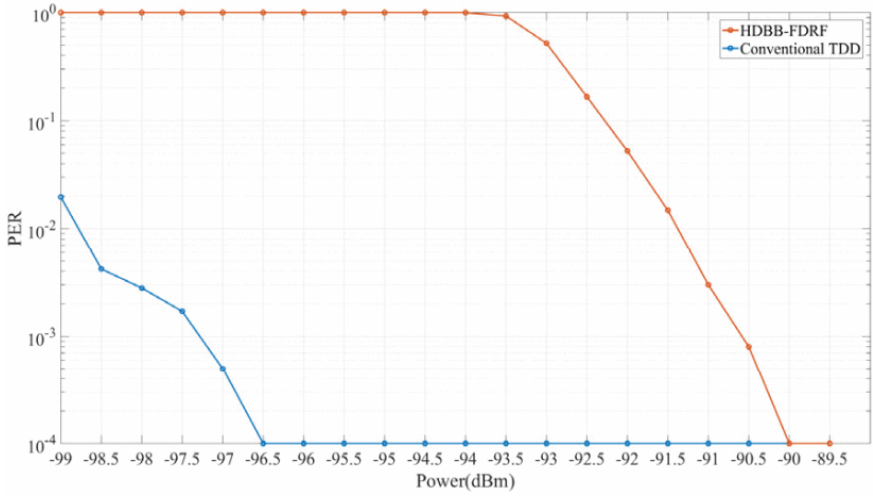


Figure 4.9: Receiver Sensitivity measurements.

well above the minimum requirement by the standard (i.e., -85 dBm at 2.4 GHz). We thus expect no severe impact on other systems.

4.5.4 Power Reduction Scheme

To reduce power consumption, a Finite State Machine (FSM) is introduced in BBHD-RFFD approach. It alleviates the power consumption by ensuring a smart switching of the SDR node among different modes. The proposed FSM illustrated

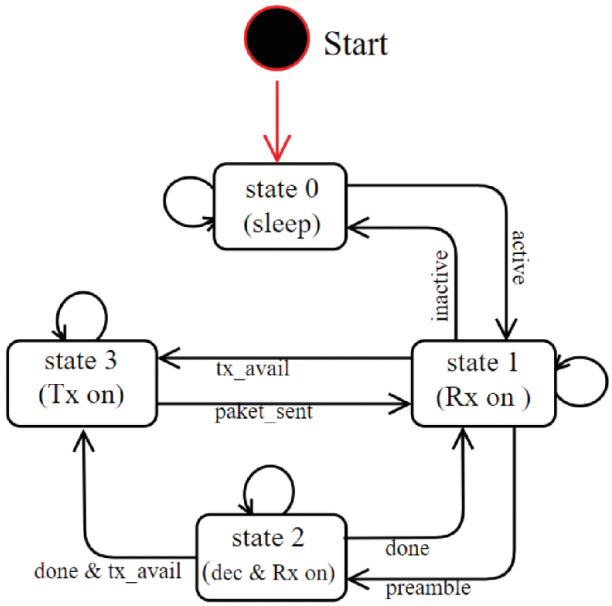


Figure 4.10: State machine diagram of operating modes used in BBHD-RFFD approach.

in Fig. 4.10 has 4 states:

1. State 0: after initialization, the SDR node enters into sleep mode, where Tx and Rx related BBPUs and analog RF front-end remain off, indicating no power consumption.

2. State 1: upon receiving active signal from the upper layer, the SDR node switches to Rx mode. The Rx mode enables the SDR node to receive a packet by turning on Rx path (i.e., both BBPUs and analog parts). Our implementation has completely separated Tx and Rx paths, and these paths can be controlled without affecting each other. Hence, Tx path (Both analog and BBPUs parts) is turned off in this mode. However, during state 1, the upper layer can issue "tx_avail" signal in case there is a packet to be transmitted. SDR node responds to this command by switching its Rx mode to Tx (state 3 in Fig. 4.10).

3. State 2: the SDR node in Rx mode continuously scans the channel and it starts decoding if there is a valid packet in the air. Upon preamble detection, SDR node changes its state from Rx to decoding state. In decoding state, the SDR node performs two actions in parallel, 1) it continues to decode the packet; 2) it starts to activate the Tx path. When decoding process is completed (indicated by "done" in Fig. 4.10), SDR node enters into state 3

if "tx_avail" request is asserted by upper layer, otherwise it returns to state 1.

4. State 3: in Tx mode the SDR node is able to transmit a packet if there is a transmit packet request from the upper layer. The FSM can enter into state 3 either via state 1 or 2. In both cases zero TT from Rx to Tx is ensured. During state 1 to state 3 transition, this can be realized by the upper layer, which can send "tx_avail" command in advance (e.g., it can send this command during packet preparation). Similarly during state 2 to 3 transition, it is achieved by activating Tx path during decoding process of state 2. Furthermore, In Tx mode, SDR node starts to activate its Rx mode at some moment during transmission and this moment is decided by transmitting packet length. The SDR node turn off its Tx mode immediately after completing the transmission. This smart scheduling makes Tx to Rx switching time (or TT) zero, because Rx switching is initiated during transmission.

Conclusively, the smart switching in the proposed FSM makes BBHD-RFFD approach feasible for real-time system where energy consumption is one of the utmost requirements.

4.6 Conclusions

SDR, consisting of programmable digital baseband processing unit and configurable RF front-end, has been widely adopted due to its flexibility and reprogrammability. However, the slow TT of an SDR RF front-end hinders the SDR from prototyping a wireless communication system in which strict latency requirement is desired. In this chapter, different operational modes of AD9361, a widely used SDR RF front-end in SDR community, are thoroughly investigated. To reduce TT on an SDR RF front-end, an innovative approach BBHD-RFFD is proposed and implemented using AD9361. Furthermore, the approach is made energy-efficient by introducing a smart switching scheme. Our approach enables researchers and wireless developers to prototype today's and future time critical communication standards on SDR platform over wide range of radio spectrum. The main challenge of using the proposed approach is to minimize the impact of self-interference. This can be done by using a power splitter with high quality isolation, or by simply placing antennas attached to Tx/Rx ports orthogonal to each other. Experimental results show that the impact of self-interference on receiver sensitivity can be limited to 7.5 dB, when suggested instructions are taken into account. The proposed approach is straightforward to use, and is generally applicable to any wireless standard, hence making SDR a powerful tool for rapid prototyping of real-life communication protocols.

Acknowledgment

This work was supported by the European Union's Horizon 2020 Research and Innovation Programme (ORCA project) under Grant 732174.

References

[1] D. Guo and L. Zhang. *Virtual full-duplex wireless communication via rapid on-off-division duplex*. In 2010 48th Annual Allerton Conference on Communication, Control, and Computing (Allerton), pages 412–419. IEEE, 2010. doi:https://doi.org/10.1109/ALLERTON.2010.5706936.

[2] *AD-FMCOMMS2-EBZ User Guide*, 2021. Available from: https://wiki.analog.com/resources/eval/user-guides/ad-fmcomms2-ebz.

[3] *USRP products*, 2022. Available from: https://www.ettus.com/products/.

[4] *User Guide of AD-FMCOMMS3-EBZ an FMC Board for the AD9361*, 2021. Available from: https://wiki.analog.com/resources/eval/user-guides/ad-fmcomms3-ebz.

[5] *User Guide of AD-FMCOMMS5-EBZ an FMC Board for the AD9361*, 2021. Available from: https://wiki.analog.com/resources/eval/user-guides/ad-fmcomms5-ebz.

[6] *802.11-2020 - IEEE Standard for Information Technology–Telecommunications and Information Exchange between Systems - Local and Metropolitan Area Networks-Specific Requirements - Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications.*, 2020.

[7] *Data Sheet of MAX2828/MAX2829 Single-/Dual-Band 802.11a/b/g World-Band Transceiver ICs*, 2004. Available from: https://datasheets.maximintegrated.com/en/ds/MAX2828-MAX2829.pdf.

[8] S. Nagata, L. H. Wang, and K. Takeda. *Industry perspectives*. IEEE Wireless Communications, 24(3):2–4, 2017. doi:https://doi.org/10.1109/MWC.2017.7955902.

[9] X. Jiao, I. Moerman, W. Liu, and F. A. P. d. Figueiredo. *Radio hardware virtualization for coping with dynamic heterogeneous wireless environments*. In International Conference on Cognitive Radio Oriented Wireless Networks, pages 287–297. Springer, 2017. doi:https://doi.org/10.1007/978-3-319-76207-4_24.

[10] *WARP v3 User Guide*, 2013. Available from: http://warpproject.org/trac/wiki/HardwareUsersGuides/WARPv3.

[11] *802.11 Reference Design for WARP v3*, 2019. Available from: http://warpproject.org/trac/wiki/802.11.

[12] B. Drozdenko, M. Zimmermann, T. Dao, K. Chowdhury, and M. Leeser. *Hardware-software codesign of wireless transceivers on zynq heterogeneous systems*. IEEE Transactions on Emerging Topics in Computing, 6(4):566–578, 2017. doi:https://doi.org/10.1109/TETC.2017.2651054.

[13] H. Wu, T. Wang, Z. Yuan, C. Peng, Z. Li, Z. Tan, B. Ding, X. Li, Y. Li, J. Liu, et al. *The tick programmable low-latency SDR system*. In Proceedings of the 23rd Annual International Conference on Mobile Computing and Networking, pages 101–113, 2017. doi:https://doi.org/10.1145/3117811.3117834.

[14] *Data sheet of AD9371 RF Agile Transceiver IC*, 2017. Available from: https://www.analog.com/media/en/technical-documentation/data-sheets/AD9371.pdf.

[15] *LabVIEW Communications 802.11 Application Framework 1.1 White Paper*, 2020. Available from: https://www.ni.com/nl-be/support/documentation/supplemental/15/labview-communications-802-11-application-framework-1-1-white-pa.html#.

[16] *Datasheet of CC2538 Powerful Wireless Microcontroller System-On-Chip for 2.4-GHz IEEE802.15.4 6LoWPAN and ZigBee Applications*, 2015. Available from: https://www.ti.com/lit/ds/symlink/cc2538.pdf.

[17] *IEEE Standard for Low-Rate Wireless Networks*, 2020. Available from: https://standards.ieee.org/ieee/802.15.4/7029/.

[18] *An Overview of Zynq-7000 SoC Data Sheet*, 2018. Available from: https://docs.xilinx.com/v/u/en-US/ds190-Zynq-7000-Overview.

[19] *Data Sheet of Mini-Circuits' ZN2PD2-63+ High-Power Splitter/Combiner*, 2022. Available from: https://www.minicircuits.com/pdfs/ZN2PD2-63+.pdf.

[20] *Data Sheet of AD9361 RF Agile Transceiver IC*, 2016. Available from: https://www.analog.com/media/en/technical-documentation/data-sheets/AD9361.pdf.

[21] *Operational Manual of MS2690A/MS2691A/MS2692A Signal Analyzer*, 2019. Available from: https://dl.cdn-anritsu.com/en-au/test-measurement/files/Manuals/Operation-Manual/MS269xA/MS269xA_SignalAnalyzer_Operation_Manual_e_33_0.pdf.

5

Hardware Efficient Clock Synchronization across Wi-Fi and Ethernet Based Network Using PTP

Chapter 2, Chapter 3 and Chapter 4 focus on hardware efficient designs for low latency wireless communication system. In addition to low latency, determinism is a prerequisite for time-critical applications. A clock synchronization algorithm is a foundation for a deterministic distributed network. This chapter presents a hardware efficient clock synchronization across wired-wireless network.

⋆ ⋆ ⋆

Muhammad Aslam, Wei Liu, Xianjun Jiao, Jetmir Haxhibeqiri, Gilson Miranda, Jeroen Hoebeke, Johann Marquez-Barja, and Ingrid Moerman.

Published in IEEE Transactions on Industrial Informatics, Vol.18, pp. 3808-3819, Oct. 2021.

Abstract Precision Time Protocol (PTP), a state-of-the-art clock synchronization protocol primarily designed for wired networks, has recently gained attention in the wireless community, due to the increased use of IEEE 802.11 Wireless Local Area Network (WLAN) in real time distributed systems. However, all the existing WLAN based PTP designs either incorporate software TimeStamping (TS)

delivering poor clock synchronization accuracy, or Hardware (HW) TS providing better synchronization accuracy at the cost of a significant amount of HW overhead. Moreover, the performance of the existing PTP solutions is mostly evaluated in single-hop wireless networks, while the performance across wired and wireless networks is taken for granted. In this chapter, a new Software Defined Radio (SDR) based approach to implement PTP is introduced and validated for IEEE 802.11 WLAN. Instead of using a dedicated HW clock, the solution utilizes the Timing Synchronization Function (TSF) clock, an existing clock in IEEE802.11 standard for synchronization between access point and WLAN stations. The performance of the proposed solution is first investigated within a single-hop WLAN and then across wired-wireless networks. Experimental results unveil that 90% of the absolute clock synchronization error falls within 1.4 μs.

5.1 Introduction

Clock Synchronization (CS) is one of the prominent technologies for real-time distributed networks. It enables the nodes in the distributed network to share the same notion of time. It is crucial for a system where performance highly depends on the CS accuracy of the networks. For example, audio or videos streaming over distributed networks, and network based motion control [1]. PTP is a state-of-the-art CS protocol introduced in the IEEE 1588 standard [2]. It is the de facto CS protocol in wired networks and is capable of providing sub-microsecond CS accuracy. The CS process in PTP is typically accomplished in two steps: (i) establishing master-slave hierarchy wherein the nodes in a network leverage the best master clock algorithm to elect a node with the best quality clock as a master clock and all the rest of the nodes' clocks are slaves; (ii) synchronizing of the master-slave clocks in which the master clock periodically exchanges special messages with slave clocks, which later extract the master clock information from these messages and synchronize to it by computing the time difference.

The aforementioned process is not only applicable to single hop network, but can be used to establish clock synchronization in a multi-hop network across different network domains (e.g., wired and wireless). A multi-hop PTP network may consist of: (i) a Grand Master (GM) clock which is the primary reference clock for CS; (ii) one or more devices with multiple PTP ports fulfilled by either a Boundary Clock (BC) or a Transparent Clock (TC); and (iii) single-port Ordinary Clock (OC) which can only be used either as slave or master clock. The main difference between BC and TC is that in TC the PTP *Sync* packet is being forwarded to downstream PTP ports without touching the origin timestamp field, instead the correction field is being updated to incorporate propagation delay and residence time inside the device, the slave PTP port then uses the correction field to update the origin timestamp; whereas in BC the origin timestamp is being updated directly

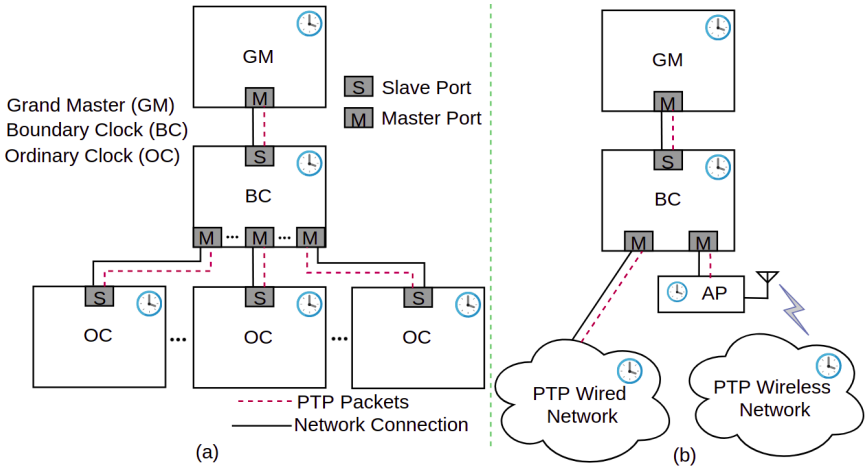


Figure 5.1: An example of PTP based (a) conventional wired network and (b) wired-wireless hybrid network when configured in E2E mode.

in the packet. Hence, BC and TC are mathematically equivalent. An example of a practical multi-hop PTP based CS network formed by BC is shown in Fig. 5.1-a.

In addition, depending on how the link delay between a PTP master and slave is measured, a multi-hop PTP network can be realized in End-to-End (E2E) or Peer-to-Peer (P2P) mode. In case of E2E mode, the slave generates requests to measure link delay towards the master; whereas in P2P mode, a device (can be a slave or bridge) generates requests towards its directly connected device, and obtain the propagation delay of only one hop. The total link delay in P2P mode is the accumulated residence time and propagation time over all the devices between a PTP slave and master. Both modes have pros and cons, P2P mode offers more accurate link delay measurement hence better CS performance, however it requires all network devices to be PTP capable; whereas E2E mode can tolerate none PTP supporting devices, but the obtained CS performance is usually worse over larger network scale, especially when non-PTP supporting devices are present.

The CS accuracy of PTP is defined as the remaining time difference between a master and a slave which can not be further reduced. The CS accuracy of PTP depends on the timestamps captured at the moment of packet reception or transmission. Fig. 5.2 shows the possible locations for TS in PTP. TS in PTP can be achieved via HW or Software (SW). HW TS is produced at or close to the physical layer, it is realized by using dedicated HW to assist the TS. On the other hand, SW TS is generated in device drivers or at a higher layer of network stack without HW assistance. The CS accuracy is significantly affected by the TS location, it is desired to place TS location as close as possible to the physical layer, so that the TS is least affected by the time variation of packets going through the network stack.

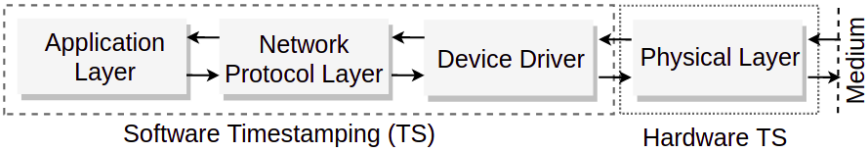


Figure 5.2: Locations for timestamping in clock synchronization protocols.

Hence, a HW TS based PTP clock is generally more accurate than a SW TS based PTP clock.

Most of the PTP based networks have been implemented over the wire [3–5]. CIPSync is a PTP compliant industrial solution which provides CS accuracy of a few nanoseconds over conventional Ethernet [4]. Another example is PROFINET, it uses PTP as CS protocol over industrial Ethernet [5]. The industry however is increasingly interested in extending the PTP from wired network to wireless in the form of a hybrid multi-hop network for more flexibility, increased scalability and reduced deployment cost [6]. An example of a simple hybrid multi-hop PTP network is shown in Fig. 5.1-b.

Academic researchers have proposed several wireless CS solutions [7–13]. SW TS based PTP solutions are easy to realize, but they provide relatively poor CS accuracy [7–9]. Regarding the realization of HW TS based PTP network, 802.1AS [14], a specific profile of PTP referred to as the generalized PTP (gPTP) is commonly used in industrial applications. 802.1AS provides approaches to maintain high precision CS across Ethernet and Wi-Fi, leveraging on specific 802.11 frames in the Timing Measurement (TM) or Fine Timing Measurement (FTM) to trigger TS in Wireless Fidelity (Wi-Fi) cards. So far only Intel claims to have working CS solution over Wi-Fi cards with this approach [15, 16], though it seems to be specific to Windows operating system and has very limited information exposed. Further exploration regarding this approach is detailed in Section 5.2. [17] uses TM frames based 802.1AS for CS over Intel Wi-Fi 5 cards in collaborative robotics applications. Though this work focuses on combining robotic operating system with wireless Time Sensitive Networking (TSN) traffic flow to maintain optimal network jitter and latency, it never quantified CS performance directly, and neither exposed any information regarding how 802.1AS is realized over Wi-Fi TM. Due to the rather rare and limited FTM/TM support in available Wi-Fi cards, most academic solutions for HW TS have not followed this approach [10–13], nevertheless the results of these work demonstrate that HW TS based PTP gives better performance, though, at the cost of dedicated HW added to realize TS. Moreover, the CS performance of these PTP solutions is mostly analysed only in a single-hop wireless network. The support of PTP BC/TC is not mentioned, therefore the performance of these solutions over a combined wired and wireless network is either taken for granted or simply left as future work.

In this chapter, we enable HW TS based PTP in *openwifi* [18], an open-source Wi-Fi chip design. Instead of using an additional clock for HW TS, we leverage the existing TSF clock with minimal modifications in Field Programmable Gate Arrays (FPGA) to achieve the same purpose. Moreover, we add necessary callbacks in the *openwifi* driver to make the TSF clock compliant to the PTP Hardware Clock (PHC) subsystem of Linux. In this way, we can use the existing PTP application and the support infrastructure in Linux [19]. Lastly, the performance of our solution is characterized over both single-hop and multi-hop network across wired and wireless network domains.

The rest of the chapter is structured as follows. The state of art of standardization, available tools and related work is detailed in Section 5.2. Section 5.3 describes the proposed method to enable PTP on IEEE 802.11 network. Experimental results are discussed in Section 5.4. Lastly, conclusions and future work are given in Section 5.5.

5.2 State of the ART

The purpose of this work is to provide a PTP based CS solution across Wi-Fi and Ethernet network. In this section, we (i) examine the relevant standardization, (ii) identify potential solutions based on the standards and available resources, (iii) present a comparison of the selected approach against existing work, and (iv) finally summarize the contribution of this chapter.

5.2.1 Standardization

The Wi-Fi alliance has introduced the *Wi-Fi TimeSync* certificate [20], which recommends to follow the IEEE 802.1AS (or gPTP) for clock synchronization over Wi-Fi. Some of the key differences between the gPTP and the original PTP protocols are that gPTP excludes usage of BC, it uses a TC to operate as a bridge; and all devices involved in gPTP network should provide HW TS and operate in P2P mode.

Unlike the original PTP protocol, 802.1AS separates the medium independent and medium dependent layers. Medium independent layer is common for all network types, which uses timestamps provided by medium dependent layer to perform CS; the medium dependent layer on the other hand is different for various network types, such as Ethernet and Wi-Fi. For Ethernet, the packets being timestamped are the event packets defined in the PTP protocol; whereas for Wi-Fi, the 802.11 MAC Layer Management Entity (MLME) is responsible for generating packets defined in TM to trigger TS in the Wi-Fi card. Later on the 802.11 standard introduces FTM, which is subsequently included in 802.1AS-rev.

FTM is capable of getting timestamps with finer resolution thanks to the averaging operation over multiple measurement iterations. Though FTM also requires more complicated handshaking to negotiate the configuration for bursts of consecutive measurements between devices, hence requires more effort to support. Apart from these differences, FTM and TM both are expected to provide timestamps that indicate the start of preamble of an Wi-Fi packet accurately, which serve as a way for Wi-Fi to support HW TS required by 802.1AS. From performance point of view, FTM is better than TM, as it has finer timestamp resolution. However from implementation and scalability point of view, FTM consumes more resources and may hinder the overall performance when a PTP device has to perform FTM with many peers. Since functionally FTM and TM are equivalent, we focus our discussion on TM and the original 802.1AS standard in subsequent sections.

5.2.2 Available Resources and Potential Solutions

In this section, we identify potential solutions and their available and missing features, focusing on: (i) the available software to realize 802.1AS over Wi-Fi, (ii) the support of 802.11 TM and PHC in the Operating System (OS), and (iii) the necessary features in Wi-Fi driver and hardware to support the operation. Regarding the software tools, we mainly explore whether the application supports 802.1AS wireless port (i.e., the interface to exchange timestamps with 802.11 TM) and the required 802.1AS roles (i.e. end stations and bridges). Regarding the support of PHC in OS, it comes down to two aspects, namely the capability to pass HW timestamp through the OS to the application, and the capability to control the hardware clock in a PTP compliant way.

gptp[1] is a gPTP daemon managed by the AVNU alliance, its development is mainly lead by Intel. The daemon is an open source implementation of PTP software stack of 802.1AS for Windows and Linux OS. Though upon further exploration, we find that the daemon does not support the 802.1AS wireless port in Linux OS. The authors in [21] attempt to run *gptp* daemon with TM as a hardware stack on Intel 8260 AC [22] in Windows OS. However, due to non-native support of PTP hardware clock in Windows OS, they were unable to measure the CS accuracy. The support of PTP hardware clock is well present in the Linux OS, its PHC subsystem allows control of PTP hardware clock, and specific socket options are provided to pass the timestamps through the OS. As TM packets are produced and consumed in the 802.11 MLME, we further explore the support of TM in mac80211 subsystem of Linux kernel, the conclusion is however that the kernel does not provide support for constructing TM packet. The interface between kernel and Wi-Fi driver only defines functions to trigger measurement and get results. In addition, the structure to return TM results does not contain timestamps, it only

[1] https://github.com/Avnu/gptp

Table 5.1: Comparison of potential ways to realize PTP across wired and wireless network, $Need$ indicates a feature is required and available, $Need^*$ indicates a feature is required but missing, and $NoNeed$ indicates a feature is not needed for an approach.

	Application		Operating System		Driver & HW		
	802.1AS Wireless Port	802.1AS all roles	PHC support	TM support	TS PTP PKT	Construct TM PKT	TS TM PKT
gptp in Windows	$Need$	$Need$	$Need^*$	$Need^*$	$NoNeed$	$Need^*$	$Need^*$
gptp in Linux	$Need^*$	$Need$	$Need$	$Need^*$	$NoNeed$	$Need^*$	$Need^*$
linuxptp 802.1AS TS TM	$Need^*$	$Need^*$	$Need$	$Need^*$	$NoNeed$	$Need^*$	$Need^*$
linuxptp E2E TS PTP [this work]	$NoNeed$	$NoNeed$	$Need$	$NoNeed$	$Need$	$NoNeed$	$NoNeed$

passes the averaged round trip time to the upper layers. This means without kernel modification, even if a Wi-Fi card supports TM, the result obtained can only be used for ranging application, not CS. In short, if we want to accomplish our goal using *gptp* daemon in Windows, we need to add hardware PHC support in Windows OS; if we proceed with *gptp* in Linux, we need to extend the *gptp* application to support wireless port and modify the Linux kernel for TM support. Both approaches already require a considerable amount of work in pure software stack, and above that the driver and hardware should be able to construct TM packet and timestamp it.

The *linuxptp* is another open source implementation of the PTP software stack in Linux OS. It is a widely used software stack in both academia and industry for CS in Ethernet based distributed systems. Though it supports both P2P mode and E2E modes, it only supports the IEEE 802.1AS in the role of end station. In other words, the default *linuxptp* cannot fulfill the role a 802.1AS bridge in a multi-hop network. In addition, *linuxptp* replies on standard PTP packets to obtain the timestamps, the software does not include possibility to leave 802.11 MLME to handle this task. Hence the support of 802.1AS wireless port needs to be added to the application. In short, if we want to accomplish our goal using *linuxptp* in a fully 802.1AS compliant way, we need to extend the application to support 802.1AS bridge and wireless port, modify the Linux kernel for passing timestamps from the 802.11 TM, and finally construct TM packet and timestamp it in driver and hardware of Wi-Fi interface.

Inspired by Ethernet based PTP solutions, one can also achieve accurate CS in multi-hop network with merely PTP packets. Although *linuxptp* does not support the 802.1AS bridge (i.e. a TC operating in 2-step P2P mode), it does support regular IEEE 1588 BC/TC. We then come up with configuring Access Point (AP) as a BC using *linuxptp* in the E2E mode, combined with TS the PTP event packets in Wi-Fi hardware and driver. BC in E2E mode is preferred for two reasons: (i) all PTP packets of different network interfaces are generated and consumed independently, no bridging is required, hence simpler in terms of network configuration; (ii) Wi-Fi AP and clients are almost always connected, using P2P mode will lead to complexity such as the support of residence time calculation but without much performance gain. As introduced previously, BC and TC are mathemati-

cally equivalent, and the difference mainly shows when non-PTP aware devices are involved, we believe the performance penalty of using E2E BC instead of P2P TC will be trivial in wireless network domain. The same conclusion applies for replacing TM packets by PTP event packets. A summary of the pros and cons of the potential solutions are given in Table 5.1, where a "$Need$" indicates a feature is required and available, a "$Need^*$" indicates a feature is required but missing, and "$NoNeed$" indicates a feature is not needed for an approach. We observe that the last approach only requires the Wi-Fi interface to TS regular PTP packets, whereas all the rest requires some level of modification in the application and OS, in addition to the TM packet construction and TS support. Although this approach is not fully 802.1AS compliant, we believe the gain of the reduced implementation effort and the convenience to achieve PTP across Ethernet and Wi-Fi with standard OS and application significantly outweigh the drawback. Hence this is the selected approach.

Apart from *linuxptp* and *gptp*, there are other software candidates, such as *ptpd* [23] and domain time II [24]. The former is a very known open source PTP daemon in Linux OS, whereas the latter is a proprietary PTP implementation for both Linux and Windows OS from Greyware. We do not consider *ptpd* because it does not support any form of HW TS. The domain time II software is also not suitable as it is none open source and there is no clear support of 802.1AS mentioned.

5.2.3 Comparison against Existing Work

There have been many attempts to implement PTP over IEEE 802.11 WLAN. These work can be categorized into SW TS based PTP, and HW TS based PTP implementations.

The metrics commonly used in literature to evaluate the performance of these solutions are the mean (μ) and standard deviation (σ) of CS error over time. In addition to these metrics, the Wi-Fi Alliance has introduced the *Wi-Fi TimeSync* certificate to specify the requirement of CS performance between multiple Wi-Fi devices [20]. The certification requires the 90th percentile of the absolute CS error (P_{90}) to be within 5.5 μs of the observed time (i.e., 120 sec). In this chapter, (5.1) is used to quantify the P_{90}

$$P_{90} = \begin{cases} i, & \text{if } f(i) = 0.9 \\ 0, & \text{otherwise} \end{cases} \tag{5.1}$$

$$\begin{aligned} f(i) &= \frac{1}{N}\sum_{n=1}^{N} 1_{C_{err}(n)} \quad where \\ 1_{C_{err}(n)} &= \begin{cases} 1, & \text{if } |C_{err}(n)| \leq i \\ 0, & \text{otherwise} \end{cases} \end{aligned} \tag{5.2}$$

where, i value denotes the P_{90} when $f(i)$ (empirical cumulative distribution function) is equal to 0.9. To check the compliance with *Wi-Fi TimeSync* certificate, We include P_{90} together with μ and σ metrics to measure the CS performance of our solution in a network.

5.2.3.1 PTP with Software Timestamping

[7] proposes a PTP solution over IEEE 802.11 WLAN with SW TS at the application layer. To mitigate the impact of asymmetric bidirectional delay of WLAN on CS accuracy, they first designed a delay filtering algorithm based on Kalman filter, and then introduced a modified Proportional Integral (PI) controller based clock servo system. [7] is validated on a Linux based embedded development board. The μ is limited to -14.24 μs with a σ of 27.65 μs. In [8], PTP SW TS is done inside the interrupt service routine of the Ath5k Wi-Fi card[2] driver of Atheros AR5xxx chipset in Linux. However, the mean CS accuracy of this work is still 6.60 μs with σ of 0.58 μs [25]. Another design of PTP over IEEE 802.11b WLAN is realized in a Linux Personal Computer (PC) [12]. They have made changes in the radio driver to achieve SW TS, leading to better CS accuracy (i.e., μ: 4.6 μs, σ:1.58 μs) when configured in AP mode. An effort is made in [9] to investigate the performance of PTP over multi-hop hybrid network. The work has however used SW TS in the wireless part of the network and the performance of their work is not tested in the presence of non-PTP background traffic.

To summarize, all these SW TS based solutions have been realized over commercial off-the-shelf WLAN chipsets, they provide decent CS accuracy (in the order of several microsecond). However, the performance is evaluated in rather simple scenario (e.g., no traffic load and mostly over single-hop network). Additionally, propagation delay asymmetry caused by different frame sizes and Modulation Coding Scheme (MCS) can potentially harm CS performance [25, 26]. Although these solutions provide important insights for PTP implementation in wireless network, we argue that they are inadequate for industrial applications.

5.2.3.2 PTP with Hardware Timestamping

[13] introduces a HW TS based PTP solution over IEEE 802.11b WLAN. The work has used a dedicated Adder Based Clock (ABC) in FPGA for HW TS. In the prototype, a customized version of the PTP protocol is used, instead of using standard PTP messages, the synchronization data from master to slave is embedded within beacons. Experimental results show that the solution has the virtues of high CS accuracy; i.e., μ is 0.24 ns with σ of 0.53 ns. Another HW TS based PTP solution is analyzed over WLAN by using an embedded processor and Programmable

[2] https://wireless.wiki.kernel.org/en/users/Drivers/ath5k

Logic Device (PLD) in [12]. To realize HW TS, they have implemented 4 hardware modules including two 64 bits counters on the PLD. The solution gives a μ of 1.1 ns with a σ of 1.76 ns, but it uses a custom Media Access Control (MAC) mechanism, which is not compatible with the IEEE 802.11 standard. Despite the good performance, the customization of PTP and Wi-Fi stack will hinder the applicability of these solutions when needed to form a PTP based CS network across Wi-Fi and Ethernet with traffic load.

5.2.4 Contribution of this Chapter

1. This solution supports HW TS, making it more accurate than SW TS based PTP in terms of CS accuracy.

2. This work uses the existing TSF clock of Wi-Fi standard as PTP HW clock, rather than adding a new HW clock, making it more economical in terms of HW footprint.

3. Existing TSF based solutions [27, 28] only apply clock offset correction during the CS process. PTP however requires both clock offset and skew correction [29]. The CS process between the slave clock (C_s) and the master clock (C_m) can be modeled by (5.3)

$$C_s(t) = S \times C_m(t) + b \tag{5.3}$$

   where S and b represent the skew and offset difference of the slave clock with respect to its master [30]. Clock offset is the relative time difference between master and slave, whereas clock skew is the relative difference in clock frequency between the master and slave's clocks. As the clock skew between master and slave remains uncorrected in the existing TSF based solutions, the clocks quickly diverge after each correction. IEEE 802.11 [31] has specified $\pm$20 ppm frequency skew for WLAN chipsets, resulting CS errors of up to $\pm$40 μs per second if only offset correction is applied every second. Thus, we introduce skew correction feature combined with clock offset correction in a TSF clock by modifying the existing TSF block of *openwifi* in FPGA.

4. The introduction of skew correction in TSF may help to further enhance the performance of the existing applications relying on synchronized TSF clocks. Similarly, the 802.11 MLME operations relying on the TSF synchronization could gain benefit from the more accurate TSF synchronization between client and AP in the same Basic Service Set (BSS). The enhanced TSF based synchronization can also be used to maintain the better coordination among APs in overlapping BSSs.

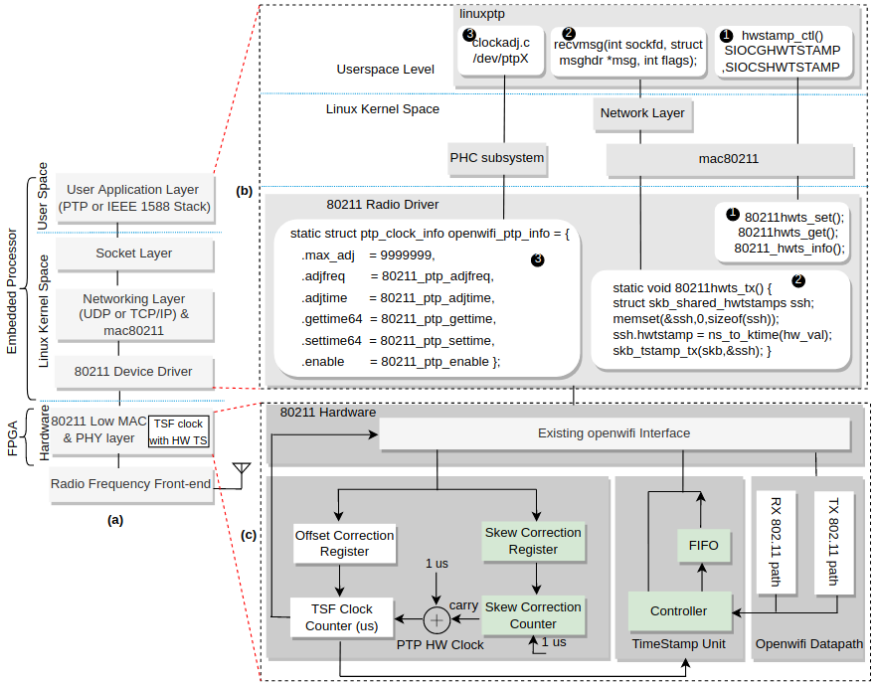


Figure 5.3: Generalized architecture (a) of our proposed PTP design over WLAN, which is primarily composed of (b) software stack and (c) hardware unit.

5. Our solution based on an open source radio chip design is full stack and Linux compatible, using existing PTP software and Linux kernel support.

6. Unlike existing work, we have quantified the CS performance in terms of P_{90} to validate the compliance of our solution with the *Wi-Fi TimeSync* certificate.

7. Lastly, to examine the adequacy of our solution for industrial applications, the CS performance is investigated over a single-hop wireless network as well as a network across wired and wireless network domains, coexisting with a large amount of non-PTP traffic.

5.3 The Proposed Solution

Fig. 5.3-a displays the block diagram of our proposed architecture for PTP design over IEEE 802.11 WLAN. The architecture is prototyped on a System on Chip (SoC) where the *openwifi* is running. PTP software stack and PTP HW clock

are realized on embedded processor and FPGA parts of the SoC, respectively. To implement the PTP software stack, we have employed the existing *linuxptp*[3] software as a userspace application. *Linuxptp* software is the Linux based PTP implementation for wired network. It relies on certain system calls to determine a device's TS capability (i.e., SW or HW TS), which are not accessible for Wi-Fi card driver, hence changes are made to bypass the check without hurting *Linuxptp*'s compliance with the PTP standard. We have further modified the radio device driver, making it able to interface with the PTP HW clock. Lastly, necessary changes are made in the existing TSF hardware to enable skew corrections, so that it can be used as a PTP HW clock. In short, the proposed PTP solution can be generalised into two steps: (1) design a wireless PTP software stack, and (2) modification of the existing TSF clock in HW to allow PTP HW TS. These two steps are detailed in this section.

5.3.1 Design of the PTP Software Stack

A diagram containing the main components used in the PTP software stack is shown in Fig. 5.3-b. There is an existing PHC subsystem in the Linux kernel. The PHC subsystem offers Application Programming Interfaces (APIs) for both userspace applications and device drivers to control the HW clock. Introducing PTP clock support for a WLAN interface requires the integration of these APIs together with TS at both userspace level and device driver.

We adopt *linuxptp* as the software stack of PTP in userspace. *linuxptp* uses the APIs of the PHC subsystem in combination with *SO_TIMESTAMPING* socket option to regulate the HW clock. *SO_TIMESTAMPING* is a socket attribute used by *recvmsg()* in userspace to generate timestamps on transmission, reception or both. The socket attribute supports both HW and SW TS. To run *linuxptp* over WLAN, it is essential that the radio driver has support for the PHC interface and *SO_TIMESTAMPING* socket option. To this end, the following steps are taken to run *linuxptp* over *openwifi* in the radio driver:

1. *linuxptp* uses *SIOCSHWTSTAMP* in ioctl system call to configure which outgoing and incoming packets should be timestamped. The packet selection is needed, as non-PTP packets on the network stacks requires no HW TS. The corresponding callback function implemented in our driver is *80211hwts_set* (see ① in Fig. 5.3-b). *SIOCGHWTSTAMP* is a system call to read the already configured packet settings and its callback function in the driver is *80211hwts_get*. *80211hwts_info* is another callback function which returns the TS capabilities of a Network Interface Card (NIC), when *ethtool*[4] *-T sdr0*

[3] http://linuxptp.sourceforge.net
[4] ethtool command: https://linux.die.net/man/8/ethtool

command is executed, where *sdr0* represents the *openwifi* interface recognized by the Linux OS.

2. After configuring the HW timestamp settings for PTP packet, the next step is to enable the HW TS in the radio driver. *sk_buff* is a data structure in Linux networking stack to handle the packet that has been received or is about to be transmitted. The structure allows the HW timestamps to be stored in the field *skb_shared_hwtstamps* optionally. A device driver is responsible for reading the timestamps from hardware registers and writing them into the *skb_shared_hwtstamps*. The function callback where the radio driver copies TS value from HW register to *sk_buff* upon packet transmission is shown in Fig. 5.3-b (see ② in radio driver).

3. The last step is to expose the HW clock to *linuxptp*, so that it can regulate the clock from userspace. The *ptp_clock* is a structure representing the PTP clock in a radio driver. It provides an abstraction on top of the HW clock and allows the userspace application to get, set and adjust the HW clock automatically. An example of the code snippet with the key functionalities of *ptp_clock* is shown in Fig. 5.3-b (see ③ in the radio driver).

   The *80211_ptp_adjtime* and *80211_ptp_adjfreq* are the most important callback functions. The former function performs clock offset correction and the latter does skew correction of PTP HW clock. The clock skew correction value is calculated using (5.4).

$$S_c = \left(\frac{10^9}{F_r \times ppb} \right) \times 10^6 \tag{5.4}$$

   Where, *linuxptp* computes the ppb (part per billion) value based on the received HW timestamps, and sends it to the radio driver. Our radio driver uses this value to calculate the S_c (skew correction) value, which corresponds to the duration in microsecond that the clock is adjusted periodically based on the ppb value. F_r denotes the desired frequency in Hz of PTP HW clock. Let's say the quantified ppb value is 78,096 and F_r value is 1 MHz, then the actual frequency of PTP HW clock becomes 1.000078096 MHz. The calculated S_c value using ((5.4)) is 12,805 μs, which is later used by PTP HW clock (see Fig. 5.3-c) to correct clock skew. Lastly, the ptp_clock is exposed as a character device ($/dev/ptpX$) to userspace, where *linuxptp* directly uses it to regulate the HW clock.

5.3.2 Design the assisting HW for PTP HW TS

As shown in Fig. 5.3-c, the assisting HW in our design primarily consists of (1) PTP HW clock representing real time local HW clock, (2) TimeStamp Unit (TSU)

Table 5.2: Hardware utilization comparison when either TSF or a dedicated clock is used as PTP HW clock.

Resources	LUTs	FFs
Default TSF	76	74
TSF as PTP HW Clock	177	130
Dedicated PTP HW Clock	154	136
HW Efficiency	23%	38%

responsible for generating HW TS upon detection of reception or transmission of frame preamble, and (3) IEEE 802.11 datapath employed for sending and receiving PTP packets. The PTP software stack uses the existing *openwifi* interface to communicate with 802.11 datapath.

Instead of using a conventional way to design a PTP HW clock, our design realizes the PTP HW clock by leveraging the existing TSF clock. The motivation of this design choice is twofold: (i) modifying the existing TSF clock costs less hardware resources than adding an additional clock; (ii) applications relying on TSF for synchronization may also benefit from this approach. The Wi-Fi standard provides basic synchronization across the TSF clocks in a BSS, though there is no rate correction and neither link delay measurement, hence the CS accuracy of the Wi-Fi standard is worse than PTP. For instance, IEEE 802.11 has specified ± 20 ppm frequency skew for WLAN chipsets, which can additionally introduce CS errors of up to ± 40 μs per second if basic TSF based CS (without rate correction) is applied every second. Despite of this fact, many researchers and developers take advantage of this feature. For instance, [32] uses the synchronized TSF in a Wi-Fi BSS for Time Division Multiple Access (TDMA), though due to the poor CS accuracy, the solution requires 20 μs guard interval in a TDMA slot. In [33], the synchronized TSF is used to coordinate transmission from a Wi-Fi BSS to avoid interfering with a sensor network, it is required to mute the Wi-Fi transmission somewhat in advance to account for the CS inaccuracy. On top of that, IEEE 802.11 standard also uses TSF for maintaining coordination between APs in overlapping BSS. With our approach, applications relying on TSF synchronization can greatly improve their performance without any additional effort. Hence we believe that the improved CS accuracy of TSF clocks in a Wi-Fi network is substantial.

While HW CS demands both clock offset and skew correction, default TSF design is only able to do offset correction. Thus, we have modified the TSF HW making it capable of performing skew correction as well (see colored blocks in Fig. 5.3-c). Table 5.2 shows a comparison of hardware utilization when the default TSF clock (i.e., without skew correction capability), the modified TSF clock supporting skew correction feature, or a dedicated HW is allocated for the PTP

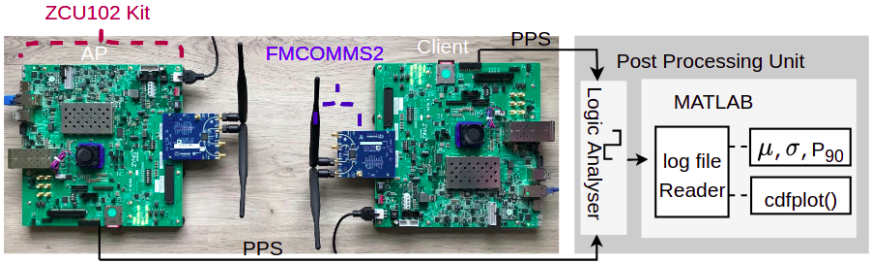


Figure 5.4: Experimental setup of the single-hop network used to measure the performance of our proposed PTP over WLAN.

HW clock. The HW efficiency in the Table 5.2 is calculated using (5.5)

$$Efficiency = \left(\frac{HW_{ptp} + HW_{dftTSF} - HW_{TSF}}{HW_{ptp} + HW_{dftTSF}} \right) \times 100 \qquad (5.5)$$

where HW_{dftTSF}, HW_{tsf} and HW_{ptp} represent the hardware consumed by the default TSF in *openwifi*, the modified TSF in our work, and a hardware clock allocated as dedicated PHC, respectively. The consumed hardware can be either Look Up Table (LUT) or Flip-Flop (FF). Table 5.2 shows that the modified TSF clock saves 23% LUTs and 38% FFs as compared to a dedicated HW PTP clock added on top of the default TSF.

The radio driver calculates the upper limit (i.e. the S_c value) for Skew Correction Counter (SCC) using ((5.4)), and loads the calculated value into the Skew Correction Register (SCREG). SCC increments with the TSF clock at 1MHz operating frequency. Upon reaching the S_c value, SCC overflows and performs the skew correction of the TSF clock. In normal situations, the TSF counter is incremented by one after each 1 μs according to the desired rate of local oscillator. Upon skew correction, the TSF counter is either incremented by two or zero depending upon the sign of carry bit. For instance, SCC performs skew correction after each 12,805 counts in the example given in the previous section.

Lastly, the TSU is composed of a controller and a First-in First-out (FIFO). The controller is in charge of executing TS upon either frame preamble transmission (or reception), and storing it into the FIFO. The radio driver later reads and copies these TS values into *sk_buff* by calling *80211hwts_tx()* function (see ② in the radio driver in Fig. 5.3-b). The radio driver also performs a similar action for reception process, for simplicity, it is not shown in Fig. 5.3-b. The radio driver performs all these actions inside the Interrupt Handler (IH). Subsequently, the *linuxptp* calls *recvmsg()* function to read these TS values and perform CS.

5.4 Results and Discussions

In this section, first, the experimental setup used to evaluate the CS performance of our design is presented. Then, the CS performance of our design is analysed over both a single-hop Wi-Fi network and across Ethernet and Wi-Fi network. The CS performance of our design is quantified in terms of μ, σ and P_{90}. Lastly, the CS performance of our design is compared against the performance of existing CS solutions. Since the solutions in literature only provides information regarding μ and σ values, we have estimated P_{90} value with the help of (5.6). We assume that the samples in a CS error measurement follows Gaussian distribution, then its absolute value will follow folded Gaussian distribution, whose cumulative distribution function is given in (5.6)

$$f(C_{err}) = \frac{1}{2}\left[erf\left(\frac{C_{err}+\mu}{\sigma\sqrt{2}}\right) + erf\left(\frac{C_{err}-\mu}{\sigma\sqrt{2}}\right)\right] \tag{5.6}$$

$$erf\left(\frac{C_{err}\pm\mu}{\sigma\sqrt{2}}\right) = \frac{2}{\sqrt{\pi}}\int_0^{\left(\frac{C_{err}\pm\mu}{\sigma\sqrt{2}}\right)} e^{-t^2}\,dt \tag{5.7}$$

where, $f(C_{err})$ reflects the estimated percentile of absolute CS error at $|C_{err}|$th point and C_{err} represents CS error between the master and the slave clock. In other words, the $|C_{err}|$ value corresponds the estimated P_{90} when $f(C_{err})$ is equal to 0.9.

5.4.1 Experimental Setup for a Single-Hop Network

To measure the performance of our proposed PTP solution with HW TS over single-hop WLAN, a wireless network between two *openwifi* SDR boards has been established, as illustrated in Fig. 5.4. The WLAN is set-up in infrastructure mode, where one SDR acts as the AP and the other behaves as a client. Due to the lack of Power Amplifier (PA) in the used Radio Frequency (RF) frontend, the SDRs are placed in close proximity (i.e., the measured transmit power of the used SDR is -15 dBm). The SDR used in the particular experiment is composed of Zynq UltraScale+MPSoC ZCU102 Evaluation Kit[5], and FMCOMMS2[6], an analog RF frontend. The Zynq SoC on ZCU102 Evalutaion Kit further comprises Programmable Logic (PL) (or FPGA) and Processor System (PS) (or ARM Cortex-A53). The PTP hardware unit along with the low MAC and physical layers of *openwifi* are implemented in the PL part, while high MAC and other layers of network stacks of *openwifi* and PTP software stack are running on embedded PS part. The OS running on the PS is Linux with kernel version 4.14. In the particular setup, the

[5]https://www.xilinx.com/products/boards-and-kits/ek-u1-zcu102-g.html

[6]https://www.analog.com/en/design-center/evaluation-hardware-and-software/evaluation-boards-kits/eval-ad-fmcomms2.html

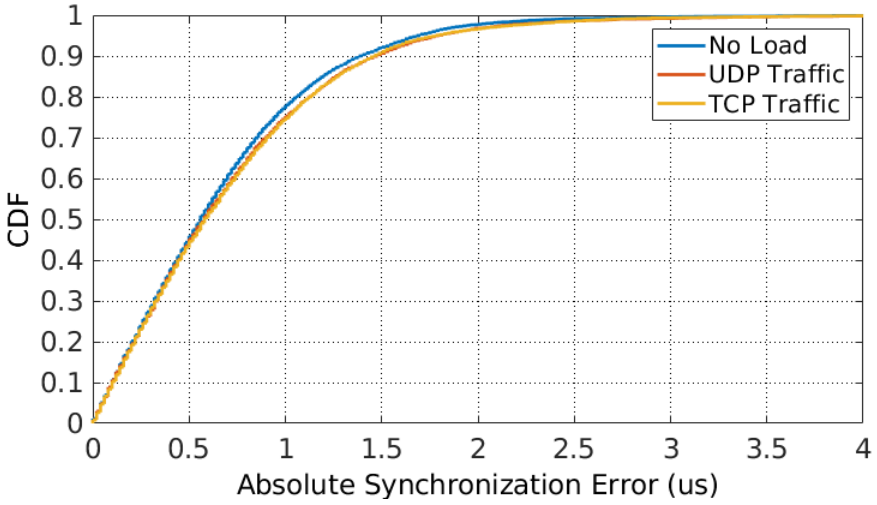


Figure 5.5: Cumulative Distribution Function (CDF) of the absolute clock synchronization error of our proposed PTP solution over single-hop WLAN when no traffic load, User Datagram Protocol (UDP) traffic and Transmission Control Protocol (TCP) traffic is applied.

openwifi is configured in IEEE 802.11a mode operating in 5 GHz frequency band with 20 MHz channel bandwidth. The MCS values for IEEE 802.11a is dynamically adapted up to 7 by Linux *mac80211* minstrel rate control algorithm.

The AP acts as the PTP master and the client is the PTP slave. In our setup, PTP is configured in E2E mode. The PTP messages in the *linuxptp* are transmitted or received using UDP/IPv4 via sockets API (see Fig. 5.3-a). The *linuxptp* v2.0 is used in the experiment, it is configured to perform synchronisation once per second. The CS accuracy is measured by analysing the Pulse Per Second (PPS) signal based on TSF clock from both slave and master with Saleae logic analyzer[7]. The sampling rate of the logic analyzer is configured at 100 MHz, resulting in 10 ns resolution. Subsequently, the PPS signals captured with the help of the logic analyser are fed to post processing unit indicated in Fig. 5.4, where the CS performance metrics (i.e. μ, σ and P_{90}) are derived based on the difference between the PPS pulses.

5.4.2 Experimental Evaluation over Single-hop WLAN

Before evaluating the CS accuracy over the single-hop WLAN, We first quantify the Convergence Time (CT) of our proposed PTP solution using experimental setup shown in Fig. 5.4. We define CT as a time when CS error or frequency offset

[7]Saleae logic analyzer https://www.saleae.com/

Table 5.3: The measured clock synchronization performance of our proposed PTP over single-hop WLAN.

Parameters	No load	UDP load	TCP load
μ	-0.279 μs	-0.330 μs	-0.325 μs
σ	0.820 μs	0.872 μs	0.868 μs
P_{90}	1.40 μs	1.48 μs	1.46 μs

(i.e., the measured clock skew between the PTP master and slave clocks in ppb) reaches a stable range which can be positive or negative depending on whether the slave clock is leading or lagging behind the master clock. Short CT is always desired, since time critical applications can run only when CS error is stable and small enough. Experimental results show that CT takes ≈4-6 master-slave interactions on the typical SDR platform with oscillator frequency error of ±61.5 ppm[8] to get stabilized.

A realistic wireless network in the industrial environment contains background traffic in addition to PTP messages which can degrade the CS performance, due to the delay or dropping of PTP packets when encounter network congestion. To this end, the CS performance is quantified when no network load is present between the SDRs to show the optimum performance, and also when network load is present between the SDRs to show the performance in a more practical condition. The measurement is conducted in an office environment, the presence of other Wi-Fi devices does not show visible impact to the experiment result. The HW setup displayed in Fig. 5.4 is leveraged to quantify the CS error. Note that the experimental evaluation uses results of the stable phase of PTP. The measurements last for 20 minutes with the synchronization performed once per second between master and slave devices.

The CS errors quantified under different network loads are shown in Fig. 5.5 and Table 5.3. As seen from Fig. 5.5, the P_{90} is 1.40 μs with σ of 0.82 μs without traffic load. The iperf[9] software is used to generate traffic between the SDRs. First a TCP stream is enabled from the client to the AP accompanied by PTP message exchange. Since TCP is by design bi-directional, using TCP of iperf will automatically find the maximum throughput (i.e. 12 Mb/s in this experiment) between the client and AP, it shows that we can push the limit of traffic load between the SDRs while keeping PTP stable. The P_{90} rises to 1.46 μs with σ of 0.87 μs while running the TCP traffic (see Fig. 5.5). Then UDP traffic is enabled in similar way, UDP is unidirectional, the average throughput found in TCP is applied as the target throughput in UDP stream. The P_{90} jumps to 1.48 μs with σ of 0.87 μs when 12 Mb/s UDP iperf is enabled. The results show that the impact

[8] https://www.xilinx.com/products/boards-and-kits/ek-u1-zcu102-g.html
[9] Iperf command: https://iperf.fr/

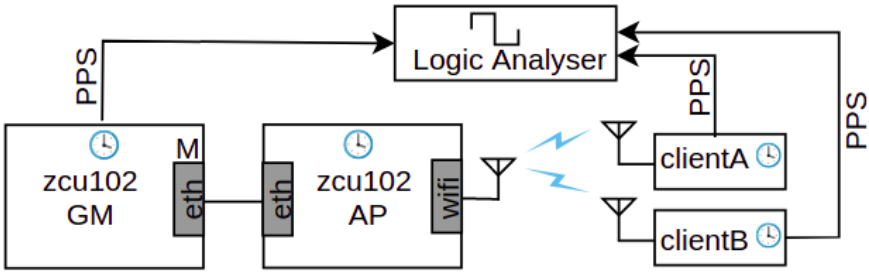


Figure 5.6: Experimental setup used to measure the performance of our proposed PTP across wired-wireless network.

Table 5.4: The measured clock synchronization performance of our proposed PTP across wired and wireless network.

Description	No load			UDP load			TCP load		
	μ (μs)	σ (μs)	P_{90} (μs)	μ (μs)	σ (μs)	P_{90} (μs)	μ (μs)	σ (μs)	P_{90} (μs)
ClientA to GM	0.94	1.42	2.88	0.98	1.49	2.98	0.90	1.65	3.16
ClientB to GM	0.87	1.43	2.81	0.93	1.42	2.82	0.87	1.68	3.10
ClientA to ClientB	-0.08	0.94	1.54	-0.05	1.02	1.66	-0.03	1.19	1.80

of network load on the CS error is almost negligible; i.e., the P_{90} only increases by 0.08 μs and 0.06 μs with UDP and TCP traffic load applied, respectively, when compared against the case when no load is applied. Last but not least, experimental results unveil that our solution is capable of providing a P_{90} much better than 5.5 μs, which is required for *Wi-Fi TimeSync* [20] certification, in both optimum and more practical scenarios.

5.4.3 Experimental Evaluation across WLAN and Ethernet

The experimental setup used to measure the CS performance across the Wi-Fi and Ethernet network is shown in Fig. 5.6. An additional Zynq UltraScale+ MPSoC ZCU102 Evaluation Kit (refferred to as ZCU102 hereafter) is used as PTP GM, as the board supports PTP HW TS and PPS on its Ethernet port. The *openwifi* AP composed of a ZCU102 and FMCOMMS2 behaves as a BC. The AP has two PTP HW clocks. The HW clock attached to Ethernet interface acts as a slave clock to the GM and the HW clock attached to the *openwifi* interface functions as the master clock to the Wi-Fi clients. Within the AP, the openwifi interface's PTP

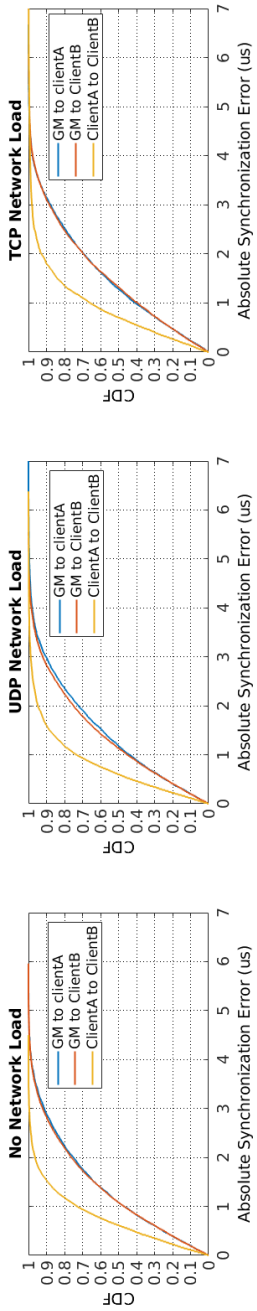


Figure 5.7: CDF of absolute clock synchronization error of our proposed PTP across wired and wireless network when no traffic load, UDP traffic and TCP traffic is applied.

HW clock is synchronized against the Ethernet's PTP HW clock, using *phc2sys* command offered in the *linuxptp* software. Thanks to the selection of E2E mode, the PTP packets are sent in layer 3 encapsulation (i.e., IPv4/UDP packets). There is no bridging of PTP packets between Wi-Fi and Ethernet, each interface is an independent PTP port that generates and consumes PTP packets in its own Internet Address (IP) subnet, which greatly simplifies the network configuration. The wireless network includes two Wi-Fi clients. Each client consists of a ZCU102 and an FMCOMMS2 board. The HW clocks of the two clients in the wireless network are slaves to the AP's *openwifi* PTP HW clock.

The duration of the measurements is set to 20 minutes with the synchronization performed once per second between PTP master and slave devices. Similar to the single-hop WLAN, the CS performance is quantified both with and without network load. For the network load setting, first two TCP iperf streams are simultaneously enabled from both Wi-Fi clients to the AP. Then the TCP streams are replaced by UDP streams, with a target throughput of 8.5 Mb/s on each client (i.e., 8.5 Mb/s is the average throughput observed in the TCP measurement).

The μ, σ and P_{90} values between the GM and Wi-Fi client across Wi-Fi and Ethernet network is shown in Table 5.4 and Fig. 5.7. Comparing to the results obtained in single-hop WLAN, the P_{90} between GM and client has increased from 1.4 μs to 2.8 μs approximately. This change is mainly caused by the additional CS error being accumulated over the extra hop (i.e., AP acting as a BC in between GM and Wi-Fi client). Intuitively, the AP clock is varied periodically as a BC, making it more challenging for the clients to synchronize, hence increasing the CS error of the downstream network. Though, similar to the conclusions from the single-hop experiment, it can be observed that the impact of network load on CS error between each client and GM is almost negligible. In addition, we also measure the difference of PPS pulses between the two Wi-Fi clients, which is much smaller than the error between a client and GM. We believe this is important for applications such as a wireless speaker or synchronized movement of robotic arms. In this type of applications, the end goal is that the actions taken on multiple PTP slaves' are synchronized. In general, even with traffic load and AP configured as a BC in between the GM and Wi-Fi clients, the P_{90} of E2E CS error is still well within the 5.5 μs required by the *Wi-Fi Timesync* certificate for performance over a single hop WLAN.

5.4.4 Comparison against the Existing PTP Solutions

A detailed CS performance comparison of the proposed PTP design with the existing CS solutions over WLAN is depicted in Table 5.5. The NA (i.e., Not Available) in Table 5.5 corresponds to a metric which is not available for a solution in the literature. The column *Sync interval* indicates the time interval in seconds af-

Table 5.5: Performance comparison with existing clock synchronization solutions of WLAN in the literature.

	HW TS	Extra HW clock	Sync interval (sec)	Clock synchronization performance over single-hop wireless network						Clock synchronization performance across wired-wireless network					
				No load			Network load			No load			Network load		
				μ	σ	P_{90}	μ	σ	P_{90}	μ	σ	P_{90}	μ	σ	P_{90}
PTP based clock synchronization solutions															
Our	√	×	1.0	-279 ns	820 ns	1.40 μs	-325 ns	868 ns	1.46 μs	866 ns	1.43 μs	2.81 μs	929 ns	1.42 μs	2.82 μs
[13]	√	√	0.1	0.24 ns	0.53 ns	**1.00** ns	NA	NA	NA	NA	NA	NA	NA	NA	NA
[12]	√	√	2.0	1.10 ns	1.76 ns	**3.40** ns	NA	NA	NA	NA	NA	NA	NA	NA	NA
[7]	×	×	2.0	-14.2 μs	27.7 μs	**51.2** μs	NA	NA	NA	NA	NA	NA	NA	NA	NA
[8]	×	×	1.0	6.60 μs	0.58 μs	**7.34** μs	NA	NA	NA	NA	NA	NA	NA	NA	NA
[12]	×	×	2.0	4.60 μs	1.58 μs	**6.63** μs	NA	NA	NA	NA	NA	NA	NA	NA	NA
[9]	×	×	1.0	NA	NA	NA	NA	NA	NA	-0.7 μs	0.64 μs	**1.52** μs	NA	NA	NA
[26]	×	×	1.0	109 ns	360 ns	**0.62** μs	316 ns	1.26 μs	**2.13** μs	NA	NA	NA	NA	NA	NA
non-PTP based clock synchronization solutions															
[27]	×	×	1.0	37.5 ms	6.80 μs	**37.6** ms	NA	NA	NA	NA	NA	NA	NA	NA	NA
[30]	×	×	1.0	6.00 μs	341 ns	**6.44** μs	21.0 μs	1.14 μs	**22.5** μs	NA	NA	NA	NA	NA	NA

ter which the synchronization procedure is repeated. For a fair comparison of our validation, we have also estimated the P_{90} (see bold values in Table 5.5) of the existing solutions using (5.6).

First, we examine the performance in the single-hop WLAN without non-PTP traffic applied. The two solutions (i.e., [13], [12]) using HW TS perform better than ours because they rely on a clock with finer resolution (i.e., a few ns in [12], 11.36 ns in [13]). As elaborated previously, we opt for the existing TSF in Wi-Fi as the PTP HW clock, which counts at 1 μs. Although this choice affects our performance, it is more hardware efficient and brings benefits to other applications relying on a synchronized TSF. A secondary factor is the *Sync interval*, [13] uses a much shorter synchronization period (i.e. 0.1 second), which means the solution makes a trade-off between performance and network overhead. Among the SW TS based solutions, [26] also outperforms ours in this condition. The authors of [26] have investigated the synchronization bias caused by the asymmetrical link delay measured when MCS used in the *Sync* and *Delay_request* packets are different. They have then carefully measured the delay in the Wi-Fi interface for PTP event packets, and use the measurement to calibrate the SW TS in the Wi-Fi driver. Although this solution achieved optimal performance for one Wi-Fi card, the calibration is hardware-dependent, and hence has to be performed for each Wi-Fi card and its corresponding driver. They also need to modify the PTP application to avoid using the default link delay measured on the fly. Our solution on the other hand uses standard PTP approach, it does not require pre-calibration and the performance is not subject to the used MCS in PTP event packets thanks to the nature of HW TS (i.e., TS happens at the start of frame).

Most solutions have not evaluated the impact of traffic load, and neither the performance beyond single-hop Wi-Fi network. The exceptions are [9, 26, 30]. [26] and [30] are the only solutions which provide information pertaining to the CS performance in the presence of network load. Unlike our solution, it is shown that the CS performance of these solutions is significantly degraded by network traffic. The CS error's σ increases from 360 ns to 1.26 μs in [26] and from 341 ns to 1.14 μs in [30].

[9] is the only solution other than ours that has evaluated CS performance across wired and wireless network segment. The CS performance of this solution is apparently better than ours in the absence of network load. The network topology of this work does resemble ours in the sense that it also has a GM in wired network, a Wi-Fi AP as a BC, and a Wi-Fi client as a PTP slave. However, further exploration unveils that the experiment setup employs expensive industrial PCs and dedicated NIC for PTP support. More particularly, they attach the *syn1588*[10] NIC on each of the devices (i.e., GM, AP and client), which costs more than 3000

[10]Syn1588 NIC https://www.oreganosystems.at/products/syn1588/hardware/syn1588r-pcie-nic

USD per unit. The *syn1588* NIC offers an adder-based clock as the PTP hardware clock running on a 25 MHz oscillator. The clock is accessed by the customized driver of the Atheros 5212 Wi-Fi card to provide SW TS on the Wi-Fi interface. Hence the superior performance of [9] is thanks to the dedicated high quality clock with very fine resolution. The work has however used SW TS in the wireless part of the network and the performance of their work is not tested with non-PTP background traffic. Though the authors do mention that the solution is susceptible to non-PTP background traffic due to SW TS. Further, all the SW TS based solutions likely to suffer from propagation delay asymmetry due to different frame sizes and MCS [25].

To summarize, despite that some of the previous work do have good performance in specific settings, these solutions either rely on customized PTP stack, Wi-Fi stack or expensive and dedicated hardware, whereas our solution offers standard compliant PTP and Wi-Fi stack with no specific hardware requirement, making it more easily adopted in a real-life industrial network. In addition, our solution shows stable performance under network traffic, and it is the first to offer experimental analysis of PTP CS performance across Wi-Fi and Ethernet when traffic load is applied.

5.5 Conclusions

In this chapter, a new approach for PTP with HW TS is proposed and verified over *openwifi*, an open-source IEEE 802.11 design on SDR. The *linuxptp* application and PHC subsystem of Linux kernel are employed as PTP software stack. Instead of adding a dedicated hardware clock, we aim to use the existing TSF timer of Wi-Fi standard as PTP HW clock, for the improved hardware-efficiency and the potentials of applications and Wi-Fi management layers relying on TSF for synchronization to benefit. The TSF timer however can only correct clock offset, hence modification is done to enable skew correction, making it qualified as a PTP HW clock. It is shown that P_{90} of our work is 1.40 μs, well below the 5.5 μs requirement of *Wi-Fi TimeSync* certificate introduced by the Wi-Fi alliance. In addition, the impact of traffic load on the clock synchronization accuracy is also investigated and proven to be insignificant. Lastly, the performance of our approach is tested across wired-wireless network and observed that the overall performance is stable and satisfactory. In the future, we consider to enhance the CS accuracy by tuning the hardware clock at finer granularity without compromising the hardware utilization. Though our novel approach for clock synchronization is validated on IEEE 802.11 standard, it is not specific for this standard. In other words, the methodology to support skew correction in an existing timer with support for PTP software stack can be applied on any wired or wireless standard incorporating embedded timer.

Our solution satisfies the performance requirement of *Wi-Fi Timesync* certificate, however it is not entirely 802.1AS compliant in the sense that we capture timestamps of regular PTP packet rather than the 802.11 Time Measurement packet, and we use a PTP boundary clock rather than transparent clock to form multi-hop PTP connections. The choice of this approach is mainly based on the maturity of available software and OS support. For future work, we are open to support 802.11 Time Measurement or Fine Time Measurement for clock synchronization when the relevant upper layer components are in place and when the performance requirement can not be reached by the current solution.

Acknowledgment

This research was funded by the Flemish FWO SBO S003921N VERI-END.com (Verifiable and elastic end-to-end communication infrastructures for private professional environments) project and the Flemish Government under the ”Onderzoeksprogramma Artificiële Intelligentie (AI) Vlaanderen” program.

References

[1] B. Chen, Y.-P. Chen, J.-M. Xie, Z.-D. Zhou, and J.-M. Sa. *Control methodologies in networked motion control systems*. In 2005 International Conference on Machine Learning and Cybernetics, volume 2, pages 1088–1093. IEEE, 2005.

[2] *IEEE 1588-2019 - IEEE Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems*, 2019.

[3] G. Gaderer, R. Holler, T. Sauter, and H. Muhr. *Extending IEEE 1588 to fault tolerant clock synchronization*. In IEEE International Workshop on Factory Communication Systems, 2004. Proceedings., pages 353–357. IEEE, 2004.

[4] *CIP Sync: an Ethernet based commercial product compliant with IEEE 1588 standard*. Available from: https://www.odva.org/technology-standards/distinct-cip-services/cip-sync/.

[5] J. Feld. *PROFINET-scalable factory communication for all applications*. In IEEE International Workshop on Factory Communication Systems, 2004. Proceedings., pages 33–38. IEEE, 2004.

[6] D. Cavalcanti, S. Bush, M. Illouz, G. Kronauer, A. Regev, and G. Venkatesan. *Wireless TSN-Definitions Use Cases & Standards Roadmap*. Avnu Alliance, pages 1–16, 2020.

[7] W. Chen, J. Sun, L. Zhang, X. Liu, and L. Hong. *An implementation of IEEE 1588 protocol for IEEE 802.11 WLAN*. Wireless networks, 21(6):2069–2085, 2015.

[8] A. Mahmood, G. Gaderer, H. Trsek, S. Schwalowsky, and N. Kerö. *Towards high accuracy in IEEE 802.11 based clock synchronization using PTP*. In 2011 IEEE International Symposium on Precision Clock Synchronization for Measurement, Control and Communication, pages 13–18. IEEE, 2011.

[9] A. Mahmood and F. Ring. *Clock synchronization for IEEE 802.11 based wired-wireless hybrid networks using PTP*. In 2012 IEEE International Symposium on Precision Clock Synchronization for Measurement, Control and Communication Proceedings, pages 1–6, 2012. doi:10.1109/ISPCS.2012.6336617.

[10] A. Mahmood, R. Exel, and T. Sauter. *Impact of hard-and software timestamping on clock synchronization performance over ieee 802.11*. In 2014 10th IEEE Workshop on Factory Communication Systems, pages 1–8. IEEE, 2014.

[11] T. Cooklev, J. C. Eidson, and A. Pakdaman. *An implementation of IEEE 1588 over IEEE 802.11 b for synchronization of wireless local area network nodes*. IEEE Transactions on Instrumentation and Measurement, 56(5):1632–1639, 2007.

[12] J. Kannisto, T. Vanhatupa, M. Hannikainen, and T. Hamalainen. *Software and hardware prototypes of the IEEE 1588 precision time protocol on wireless LAN*. In 2005 14th IEEE Workshop on Local & Metropolitan Area Networks, pages 6–pp. IEEE, 2005.

[13] R. Exel. *Clock synchronization in IEEE 802.11 wireless LANs using physical layer timestamps*. In 2012 IEEE International Symposium on Precision Clock Synchronization for Measurement, Control and Communication Proceedings, pages 1–6. IEEE, 2012.

[14] *IEEE Standard for Local and Metropolitan Area Networks - Timing and Synchronization for Time-Sensitive Applications in Bridged Local Area Networks*. IEEE Std 802.1AS-2011, pages 1–292, 2011. doi:10.1109/IEEESTD.2011.5741898.

[15] *Intel. Wi-Fi 6 ax200 module*. Available from: https://www.intel.com/content/dam/www/public/us/en/documents/product-briefs/wi-fi-6-ax200-module-brief.pdf.

[16] *Intel. Wi-Fi 6 ax201 module*. Available from: https://www.intel.com/content/dam/www/public/us/en/documents/product-briefs/wi-fi-6-ax201-module-brief.pdf.

[17] S. Sudhakaran, V. Mageshkumar, A. Baxi, and D. Cavalcanti. *Enabling QoS for Collaborative Robotics Applications with Wireless TSN*. In 2021 IEEE International Conference on Communications Workshops (ICC Workshops), pages 1–6. IEEE, 2021.

[18] J. Xianjun, L. Wei, and M. Michael. *open-source IEEE802.11/Wi-Fi baseband chip/FPGA design*. Available from: https://github.com/open-sdr/openwifi.

[19] R. Cochran and C. Marinescu. *Design and implementation of a PTP clock infrastructure for the Linux kernel*. In 2010 IEEE International Symposium on Precision Clock Synchronization for Measurement, Control and Communication, pages 116–121. IEEE, 2010.

[20] *Wi-Fi Certified Timesync Technology Overview*, 2017.

[21] P. Chen and Z. Yang. *Understanding Precision Time Protocol in Today's Wi-Fi Networks: A Measurement Study*. In 2021 USENIX Annual Technical Conference (USENIX ATC 21), pages 597–610. USENIX Association, July 2021. Available from: https://www.usenix.org/conference/atc21/presentation/chen.

[22] *Intel. Dual Band Wireless AC 8260.* Available from: https://www.intel.com/content/dam/www/public/us/en/documents/product-briefs/dual-band-wireless-ac-8260-brief-v2.pdf.

[23] *PTPd source code documentation*. Available from: http://ptpd.sourceforge.net/doc.html.

[24] *Greyware. Domain Time II Performance Series Time Synchronization*. Available from: https://www.greyware.com/software/domaintime/.

[25] A. Mahmood, R. Exel, H. Trsek, and T. Sauter. *Clock synchronization over IEEE 802.11—A survey of methodologies and protocols*. IEEE Transactions on Industrial Informatics, 13(2):907–922, 2016.

[26] A. Mahmood, R. Exel, and T. Sauter. *Delay and jitter characterization for software-based clock synchronization over WLAN using PTP*. IEEE Transactions on industrial informatics, 10(2):1198–1206, 2014.

[27] A. Mahmood, G. Gaderer, and P. Loschmidt. *Software support for clock synchronization over IEEE 802.11 wireless LAN with open source drivers*. In 2010 IEEE International Symposium on Precision Clock Synchronization for Measurement, Control and Communication, pages 61–66. IEEE, 2010.

[28] A. Mahmood, R. Exel, and T. Bigler. *On clock synchronization over wireless LAN using timing advertisement mechanism and TSF timers*. In 2014 IEEE International Symposium on Precision Clock Synchronization for Measurement, Control, and Communication (ISPCS), pages 42–46. IEEE, 2014.

[29] H. Puttnies, P. Danielis, A. R. Sharif, and D. Timmermann. *Estimators for Time Synchronization—Survey, Analysis, and Outlook*. IoT, 1(2):398–435, 2020.

[30] A. Mahmood, R. Exel, and T. Sauter. *Performance of ieee 802.11's timing advertisement against synctsf for wireless clock synchronization*. IEEE Transactions on Industrial Informatics, 13(1):370–379, 2016.

[31] *802.11-2016 - IEEE Standard for Information technology—Telecommunications and information exchange between systems Local and metropolitan area networks—Specific requirements - Part 11:*

Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications, 2016.

[32] Y.-H. Wei, Q. Leng, S. Han, A. K. Mok, W. Zhang, and M. Tomizuka. *RT-WiFi: Real-Time High-Speed Communication Protocol for Wireless Cyber-Physical Control Applications*. In 2013 IEEE 34th Real-Time Systems Symposium, pages 140–149, 2013. doi:10.1109/RTSS.2013.22.

[33] M. Chwalisz and A. Wolisz. *Towards efficient coexistence of IEEE 802.15.4e TSCH and IEEE 802.11*. In NOMS 2018 - 2018 IEEE/IFIP Network Operations and Management Symposium, pages 1–7, 2018. doi:10.1109/NOMS.2018.8406146.

6
Conclusion

"That is what learning is. You suddenly understand something you have understood all your life, but in a new way."

–Doris Lessing (1919 - 2013)

To address the strict Quality of Service (QoS) requirements of existing and emerging use cases in consumer as well as industrial verticals, modern wireless communication systems are continuously evolving by introducing new technologies and features in communication standards and protocols. The realization of these new technologies and features demands complicated Hardware (HW) designs and hence lead to larger HW footprints for the Baseband Processing Units (BBPUs) of the corresponding radio Transceivers (TRXs). Thus, there is a need of a digital hardware solution that enables efficient HW implementation of the TRX, and at the same time supports flexible, reconfigurable and programmable features. The state-of-the-art Software Defined Radios (SDRs) are the attractive platforms for designing the radio TRXs of emerging wireless communication standards, customized or proprietary solutions target professional high-end solutions, as the platforms are not only flexible, but also support advanced digital processing capabilities. This PhD dissertation focuses on HW efficient designs for wireless communication systems using SDR. More specifically, the solutions presented in this PhD research target the applications, which have strict QoS requirements in terms of latency and determinism. Subsequently, These solutions have been verified for Institute of Electrical and Electronics Engineers (IEEE) 802.11 and IEEE 802.15.4 standards in real-life experiments. This chapter gives a brief summary

of the main contributions and achievements, as well as the future direction how the work can be further enhanced or extended for other standards or future high-performance wireless technologies.

6.1 Summary

Industrial IoT (IIoT) applications demand low latency and a deterministic behaviour of the communication link at a variety of data rates. In an IIoT (or Internet of Things (IoT)) network, a Gateway (GW) is the sole way of connecting the end-devices with the internet and is mostly equipped with a single User (SU) TRX. Such a GW restricts the performance of an IIoT network in a congested environment. Upgrading the GW from a SU TRX to a Multi User (MU) TRX is the most obvious way to solve the bottleneck. Many wireless communication standards have recent advances regarding MU communication. For instance, IEEE 802.11ax leverages MU-Orthogonal Frequency Division Multiple Access (OFDMA) and MU-Multiple Input Multiple Output (MIMO) technologies to concurrently serve MU communications. The demand for MU functionality poses new challenges in the wireless chip design, in particular for the baseband processing. On top of that, recent wireless communication standards used in IIoT wireless communication such as IEEE 802.11ax and 5th Generation (5G) operating in Time Division Duplex (TDD) have very strict latency requirements when switching from Transmitting (Tx) to Receiving (Rx) mode and vice versa. However, the slow Turnaround Time (TT) of the analog Radio Frequency (RF) front-ends of the state-of-the-art SDR platforms seriously restricts the design of such wireless communication standards. In addition to low latency, a deterministic link behaviour is an essential requirement for time-critical IIoT applications, which is hard to satisfy over wireless network links, due to wireless channel impairments (e.g., multipath, fading, and co-channel interference) requiring the need for link adaptation protocols (e.g., dynamic Modulation Coding Scheme (MCS) adaptation) and retransmissions due to frame errors, both leading to variable transmission times. Apart from that, many wireless protocols use non-deterministic channel access mechanisms (e.g., CSMA, ALOHA) for transmission.

The contributions of this PhD are as follows: (1) providing HW efficient architecture for the MU TRX using a SDR platform, (2) enabling the SDR for prototyping high performance of state-of-the-art and emerging wireless communication standards by reducing the TT of its RF front-end, and (3) introducing a HW efficient design for a Clock Synchronization (CS) algorithm, which is the foundation for a deterministic network (e.g., Time Division Multiple Access (TDMA) based network, Time Sensitive Networking (TSN)).

6.1.1 Hardware Efficient Designs for MU TRXs

MU communication can be divided mainly into two categories. In the first category, the capacity of the system increases with the number of users, whereas in the second category the capacity of the system remains fixed irrespective of the number of users. Concurrent multi-channel communication and MU-MIMO exploiting spatial diversity are examples of the former type of the MU communication, whereas MU-OFDMA belongs to the latter type of the MU communication.

Currently, Hardware Duplication (HD) approach is used to design MU TRX wherein a dedicated SU TRX is utilized for each user. Such an approach is obviously HW inefficient, as the HW footprint increases proportionally with the number of users simultaneously served by the MU TRX. In this PhD dissertation, a novel HW efficient approach is introduced that uses the same SU TRX HW for the MU TRX by employing the Hardware Virtualization (HV) technique. HV is a type of HW resource sharing technique whereby multiple logical instances are created from a single physical instance implemented on HW using time division multiplexing. The HV relies on multitasking, which requires a context switching mechanism. The context switching mechanism inherits the context save-restore problem. To solve this problem, an extra Finite State Machine (FSM) and a memory unit are introduced. During each context switching, the FSM is responsible for proper functioning of the MU TRX and the memory unit is used to store the internal states of the TRX of each user such as MCS value and packet length per user. Efforts are made to keep the extra HW overhead used to manage the context switching as minimum as possible.

The new HW efficient approach is first validated for an IEEE 802.15.4 complaint concurrent multi-channel TRX, which can independently transmit/receive data on up to 8 channels, on an SDR platform. The validation results show that the virtual TRX holds the same performance in terms of Receiver (RX)'s sensitivity and quality of transmitted signal (i.e., spectral mask) as the HD based TRX on different channels, but saves 82.63% Flip-Flops (FFs) and 67.15% Look Up Tables (LUTs) hardware for the particular case of 8 channels. Later, the HV concept is applied to design an IEEE 802.11ax compliant OFDMA based MU TRX. The proposed design is validated by implementing an OFDMA based MU Transmitter (TX) on an SDR platform, which can transmit up to 4 users simultaneously. Unlike multi-channel TRX, the validation is more challenging as IEEE 802.11ax compliant OFDMA includes advanced features such many MCS values, different Resource Unit (RU) sizes for each user, various Protocol Data Unit (PPDU) types. Apart from that, the Physical (PHY) layer of OFDMA based MU TX consists of more modules with different modules are required to have different processing load during a single context switching operation making the design more complicated (especially the FSM responsible for correct functioning of context switching). The validation results unveil that the OFDMA based MU TX outperforms the

requirement specified (in term of Error Vector Magnitude (EVM), spectral mask and unused tone error) in the IEEE 802.11ax standard, at the same time it saves 66.56% LUTs and 61.13% FFs hardware resources for the specific case. The validation results in both cases show that the proposed HV based approach is highly efficient in terms of hardware utilization, hence could effectively reduce the HW footprint of a final commercial chip.

6.1.2 Reducing TT of a Modern SDR

The slow TT of the analog RF front-ends of state-of-the-art SDR platforms is inadequate to design today's state-of-the-art and future high-performance wireless communication standards operating in TDD mode. Thus, a novel solution is presented in this book to reduce the TT on an SDR. This is achieved by operating the BBPU and the analog RF front-end of an SDR in half-duplex mode and full-duplex mode, respectively. The main challenge of using the novel solution is to minimize the impact of self-interference, which is mitigated by using a power splitter with high quality isolation, or by simply placing antennas attached to Tx/Rx ports orthogonal to each other. The experimental validation of the solution on the widely adopted AD9361 RF front-end shows that for any type of wireless communication standard, the TT in TDD mode can be reduced to zero by the proposed solution, with the impact of self-interference on the RX's sensitivity can be limited to 7.5 dB, when suggested recommendations are taken into account.

6.1.3 High Precision Clock Synchronization

Precision Time Protocol (PTP) is the de facto CS protocol in wired (or Ethernet) networks and provides foundation for time sensitive networking. Extending TSN capability over wireless (i.e., Wireless TSN (WTSN)) is a next step highly desired by many industries. Today, WTSN over IEEE 802.11 and 5G is in its early stages. Investigation of PTP in the wireless domain is the first and most crucial step towards WTSN. The CS performance of PTP relies on how accurate the moment of packet reception or transmission is timestamped. Existing PTP designs over wireless network either use software TimeStamping (TS) approaches resulting in poor CS performance, or HW TS providing better synchronization performance, however at the cost of a significant amount of HW overhead. Therefore, as a final contribution of this PhD dissertation, a novel HW TS based PTP solution using SDR is presented, which leverages the existing clock in a wireless communication standard as PTP HW clock for the CS purpose. While the PTP protocol requires both offset correction and skew correction for its proper functioning, the native clocks of wireless communication standards generally support only offset correction. Hence an extension to the IEEE 802.11 Timing Synchronization Function (TSF) clock is proposed for enabling skew correction. It is experimentally

validated that the native clock with skew extension is qualified as a PTP HW clock and saves significant HW resources. That is, the modified TSF clock saves 23% LUTs and 38% FFs as compared to a dedicated HW PTP clock added on top of the default TSF. Microsecond level synchronization accuracy has been demonstrated first within a single-hop Wireless Local Area Network (WLAN) (i.e., 90% of the absolute CS error falls within 1.4 μs) and then across wired-wireless networks (i.e., 90% of the absolute CS error falls within 2.8 μs).

The results achieved during the scope of this PhD can be very useful for the broader research community as the capability to use HW more efficiently on state-of-the-art SDR platforms may boost innovation of advanced features of upcoming wireless communication standards or new technologies, specifically targeting high-end solutions for professional markets such as industrial applications. Many research on new advanced radio concepts are still theoretical concepts due to lack of high-performance real-time evaluation platforms.

6.2 Future Work

Considering the targeted problems and offered solutions in this PhD dissertation, there are a number of future works that can improve the achieved results.

In this PhD dissertation, the HV technique is leveraged to design the BBPUs of an IEEE 802.15.4 based 8 channel TRX and an IEEE 802.11ax compliant MU OFDMA TRX in Chapter 2 and Chapter 3, respectively. The proposed methodology is generic, follows a user-friendly implementation strategy and can therefore be applied on any existing or future wireless standards involving MU communication or requiring HW duplication. Any MU TRX can be easily realized by executing the following steps: (1) Drawing the modular design of an MU TRX. (2) Identifying the modules for which the HW footprint increases with the number of users in the MU TRX. (3) Applying HV on the identified modules to keep the HW footprint of MU TRX as close as possible to SU TRX. Application of HV on MU TRX involves the identification of the signals from the original design that need to be saved and restored during each context switching operation, and then addition of an extra FSM and Random Access Memory (RAM) for context save-restore operations. Thus, the proposed method can be applied to other TRX or standards involving MU communication such as IEEE 802.16, Long Term Evolution (LTE), and 5G and deserves further attention, in particular in view of the European chip act[1], where the European Commission targets to bring tools, skills and technological capabilities back the Europe, the knowledge generated on HV can lead to a competitive advantage.

Chapter 4 proposed a method to reduce the TT on an SDR by operating the RF front-end in full-duplex mode and the BBPU in half-duplex mode. The proposed

[1] https://ec.europa.eu/commission/presscorner/detail/en/ip_22_729

approach is straightforward to use, and is generally applicable to any wireless standard, hence making SDR a powerful tool for rapid prototyping (or designing) of real-life communication protocols. Though the RF parts of both RX and TX remain activated in the proposed approach, which adversely affect the sensitivity of the RX. The self-interference leakage still exists in the proposed approach despite of using a power splitter with high quality isolation, or by placing antennas attached to Tx/Rx ports orthogonal to each other. One way to improve the proposed approach is to operate the RF front-end of an SDR in half-duplex mode. In the current condition, the RF front-end operating in half-duplex mode is not qualified for a solution requiring short TT, as several components (e.g., frequency synthesizer, Local Oscillator (LO) and mixer) in the RF front-end are powered up and stabilized during each TT, which consumes a considerable amount of time. One straight forward way to improve the TT could be the identification and optimization of the critical analog components in an RF front-end. For instance, in AD9361 RF front-end, the Tx to Rx TT is 4.2 μs, and Rx to Tx TT is 17.2 μs including the power-up time of Digital-to-Analog Converter (DAC), which is $\approx$17 μs. It is obvious that optimizing the power-up time of DAC can significantly improve the TT value of AD9361 RF front-end. TT could also be improved by keeping some of the analog components having higher power-up time in the RF front-end on, while others can be switched off during the TT in such way that minimize the impact of self-interference on RX's sensitivity as much as possible. One option could be to switch on/off the LO component of RF front-end during each TT, keeping the rest of the components on all the time. Hence further research on the exact component to be optimized, and how to execute such kind of optimization in the SDR RF front-end is a natural next step to improve the TT and TRX performance.

The PTP based CS is validated on the IEEE 802.11 standard in Chapter 5. However, it is not specific for this standard. In other words, the proposed methodology to support skew correction in an existing clock to be compliant with the PTP software stack can be applied on any wired or wireless standard incorporating an embedded clock. Further more, the achieved 1.4 μs CS accuracy of the proposed solution verified for IEEE 802.11 is due to the use of TSF as PTP HW clock, and TSF has a resolution of 1 μs. Since the TRX needs to run at much higher clock speed than 1 MHz to meet the required baseband sample rate and the TSF clock is incremented when a counter overflows. In the future, we can use the extra resolution provided by the counter to tune the HW clock at finer granularity for gaining the CS accuracy, without increasing the HW footprint.

A

An Enhanced Version of IEEE 802.15.4 Standard Compliant Transceiver Supporting Variable Data Rate

Chapter 2, Chapter 3, Chapter 4, and Chapter 5 focus on hardware efficient designs for low latency and deterministic wireless communication systems by making use of flexible and advanced digital processing features of the state-of-the-art Software Defined Radio (SDR). Thanks to the flexible feature of an SDR, the SDRs, Now a days, can be used to design wireless communication standards (or protocols), which provide optimal performance in a diverse radio spectrum such as in Industrial, Scientific and Medical (ISM) band. In this Appendix, an SDR based IEEE 802.15.4 compliant flexible Hardware (HW) design is introduced, which provides run-time configurable data rate and Bandwidth (BW) according to the spectrum conditions.

⋆ ⋆ ⋆

Muhammad Aslam, Xianjun Jiao, Wei Liu, and Ingrid Moerman.

Published in EuCNC 2019 : Enabling smart connectivity : proceedings., 18-21 June 2019. p. 1-2. Valencia, Spain.

Abstract In Industrial Wireless Sensor Networks (IWSNs), standard compliant state-of-the-art devices typically provide fixed data rate (for instance, Institute of Electrical and Electronics Engineers (IEEE) 802.15.4 based devices has a 250 kbps with fixed Bandwidth (BW) of 2 MHz in 2.4 GHz.). The fixed BW restricts these devices from presenting their optimal performance in a continuously varying radio spectrum environment, i.e., narrow-band modes fit best in a crowded spectrum, whereas wide-band modes perform better for low latency scenario. The flexible feature of an Software Defined Radio (SDR) allows us to propose a communication system capable of operating in both standard BW and non-standard BW modes with the same hardware accelerator. In this work, IEEE 802.15.4 compliant SDR Transceiver (TRX) is introduced, where flexible Media Access Control (MAC) and Physical (PHY) layers are implemented in Time-Annotated Instruction Set Computer (TAISC) and FPGA, respectively. This unique introduction of flexibility at both layers enables the proposed solution to operate in multiple modes which can be categorized into the three subtypes depending on the signal bandwidth, i.e. Narrow-Band (NB), standard compliant, and Wide-Band (WB). The multi-band feature enables the solution to work efficiently in a diverse radio spectrum environment. Experimental results unveil that the proposed TRX provides Receiver (RX) sensitivity of -107 dBm, -98 dBm, and -90 dBm when it is configured in NB, standard, and WB modes respectively.

A.1 Introduction

Wireless Sensor Networks (WSNs) have been increasingly used in a wide range of applications, including health care monitoring, area monitoring, home automation, and water quality monitoring. In a typical WSN, nodes (mainly sensors) are spatially distributed and are wirelessly connected to transfer data to a central location. A WSN broadly consists of three components: sensor nodes, gateways and users. The sensor nodes and gateways, which are wirelessly interconnected, may or may not be connected to the internet[1] [1]. General requirements of a WSN include small form factor, low cost, power efficiency, self-healing, scalability, and robustness. Industrial Wireless Sensor Network (IWSN) is an industrial application of WSN which requires additional requirements of link reliability, deterministic latency, and resistance to noise. Zigbee, WirelessHART, ISA 100.11a, and 6LoWPAN protocols [2–5], which are specially designed for IWSN applications, are based on IEEE 802.15.4 standard [6]. The standard defines Media Access Control (MAC) and Physical (PHY) layers specifications, which imply that all these protocols have the same MAC and PHY layers. In contrast to the upper layers,

[1] A WSN is often used within an Internet of Things (IoT) system and can be thought of as a large set of sensor nodes in the IoT system. The main function of WSN is to gather and relay sensory data through a gateway to the Internet in an IoT system.

MAC and PHY layers are generally implemented in hardware which limits the flexibility, adversely affecting the performance of a communication system. For instance, the fixed data rate (i.e., 250 kbps with a Bandwidth (BW) of 2MHz) of IEEE standard in 2.4 GHz band prevents a wireless device from obtaining desired performance (e.g., link reliability) in a continuously changing Radio Frequency (RF) environment. This is because 2.4 GHz ISM band is generally heavily occupied, fixed BW limits the capability for a device to take advantage of leftover spectrum or time gap in real time communication.

Software Defined Radio (SDR) is a newly introduced technology that enhances the flexibility of a communication system by implementing the MAC and PHY layers in software. An SDR typically consists of a programmable digital baseband processing unit(s) implemented on programmable devices, e.g., Field Programmable Gate Arrays (FPGA), Digital Signal Processor (DSP) etc., and a configurable analog RF front-end that generally includes Digital-to-Analog Converter (DAC), Analog-to-Digital Converter (ADC), filters, mixture, and amplifier. Flexible characteristic of an SDR motivates us to implement an enhanced version of IEEE 802.15.4 standard that has flexible data rate and respective BW. Data rate refers to the throughput in PHY layer, while BW refers to the signal's BW in RF channel. This flexible data rate feature together with a channel sensing mechanism drives the proposed solution to seamlessly change its BW according to spectrum availability, thus providing deterministic latency and reliability in a dynamic RF environment.

A.2 Motivation

Unlike wired communication, a wireless environment in wireless communication is highly unpredictable and dynamic. Therefore, a communication system is expected to run-time adapt according to a varying wireless environment, e.g., switching channels in accordance with the radio spectrum occupation. As an illustration (see Fig. A.1), one of the operating frequency bands of the IEEE 802.15.4 standard is unlicensed (i.e., 2.4 GHz band), making it highly susceptible to interferences from other devices. Fig. A.1 illustrates three different use cases of co-existence of IEEE 802.15.4 compliant with other standard devices.

1. Existing IEEE 802.15.4 based state-of-the-art solutions of WSN has fixed data rate (e.g., 250 kbps of IEEE 802.15.4 in 2.4 GHz spectrum) and exhibit inefficient performance in terms of reliability and latency in crowded spectrum scenario, because it may require many attempts for successful transmission. In this scenario, even channel-sensing mechanisms fail in improving performance due to unavailability of free channels. On the other hand, given the same transmit power, a low data rate communication system occupies a relatively narrow band in frequency domain, and shows a more

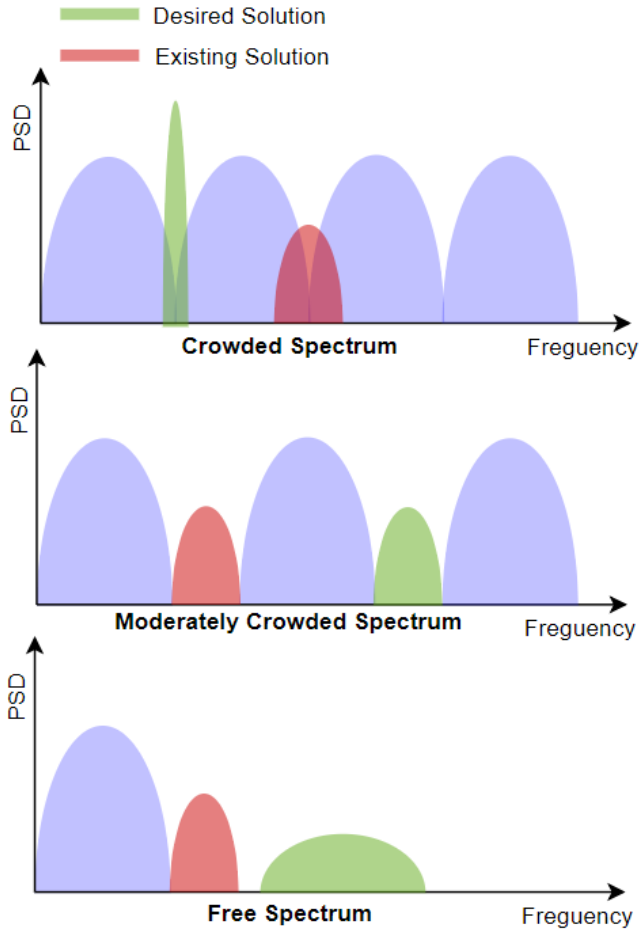


Figure A.1: Various scenarios of spectrum occupancy.

reliable communication link even in a fully crowded spectrum scenario (see Fig. A.1).

2. Similarly, devices with fixed data rate also show room for improvement when the spectrum is rather clean. In this condition, a high data rate communication can be achieved by occupying a relatively wide band in the frequency domain, exhibits a low latency communication link in a clean spectrum scenario (see Fig. A.1).

Conclusively, to obtain optimal performance in diverse RF spectrum, a smart solution with flexible data rate is desired . Therefore, we introduce an enhanced IEEE 802.15.4 compliant SDR architecture providing run-time configurable data rate

and configurable BW according to the spectrum conditions. This smart approach together with channel-sensing mechanisms outperforms the existing fixed-data rate solutions.

A.3 Related Work

The related works are categorized into two sub-groups: commercially available solutions and SDR based solutions. The following subsections thoroughly explain these sub-categories.

A.3.1 Commercially Available Solutions

There are already many solutions available in the commercial market which supports both standard compliant mode (i.e., fixed data rate and BW) and non-standard mode (i.e., variable data rate and configurable BW) [7, 8]. CC1312R [7] supports a wide range of data rate and corresponding BWs. This chip can be configured into two proprietary modes: IEEE 802.15.4g compliant mode, and the proprietary radio mode. In the proprietary radio made, CC1312R supports various data rate ranges from 625 kbps to 4 Mbps with each data rate has different BW. However, this commercial chip only operates in the Sub-1 GHz frequency band. CC1352P [8] has three modes of operations which are IEEE 802.15.4 compliant, Bluetooth 5 low energy complaint and proprietary radio modes. This chip can operate in 2.4 GHz, and Sub-1GHz bands with a data rate support up to 2 Mbps. To comply with IEEE 802.15.4 and Bluetooth, the chip provides a fixed data rate in 2.4 GHz band, whereas flexible data rate feature is achievable in Sub-1 GHz band.

A.3.2 SDR Based Solutions

There are many IEEE 802.15.4 based SDR solutions reported in the literature. Thomas et al. has implemented the first SDR architecture for the PHY layer of this standard, including dedicated Transmitting (Tx) and Receiving (Rx) chains [9]. To this end, the author has used USRP [10] and GNU Radio [11], a free & open-source software development toolkit that provides signal-processing blocks to implement software radios. This work has permitted researchers, for the first time, to access and explore the PHY layer features of IEEE 802.15.4 standard. In [12], a communication stack from PHY up to network layer is implemented on GNU Radio and USRP N210. Although both of the above-mentioned SDR-based Transceivers (TRXs) introduce considerable flexibility in the design, they provide fixed data rate, i.e., 250 kbps. The authors in [13] go one-step further and propose an efficient system that incorporates three different data rates which are 20 kbps, 40 kbps, and 250 kbps. Though the design only allows the data rate to be

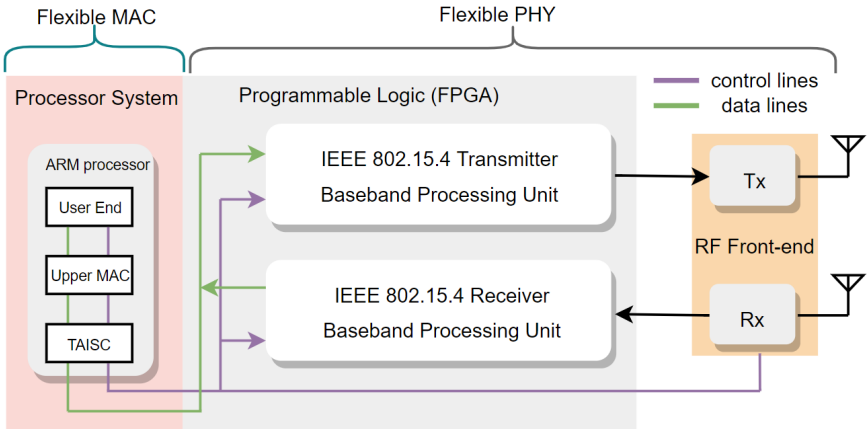


Figure A.2: A block diagram of the proposed TRX.

configured according to its operating frequency band, for example, 250 kbps data rate only works in 2.4 GHz band. Another disadvantage of this architecture is that each data rate implementation requires a dedicated hardware thereby consuming more FPGA resources.

In this work, we support not only variable data rates, but also utilize the same hardware for various data rate implementation. Furthermore, we do not have restrictions of data rate in different frequency bands.

A.4 The Proposed Solution

This section describes the proposed design for IEEE 802.15.4 based TRX capable of run-time switching between standard compliant (i.e. 250 kbps) and non-standard modes (i.e. higher or lower than 250 kbps). The block diagram of the TRX is shown in Fig. A.2. The TRX mainly consists of flexible MAC and PHY layers.

A.4.1 The Flexible Design for the PHY Layer of IEEE 802.15.4

IEEE 802.15.4 standard specifies three different PHY modes (i.e., 20 kbps, 40 kbps, 250 kbps) where each mode operates in a particular frequency band. To implement our TRX, we have chosen the PHY layer of data rate 250 kbps because of following two reasons:

1. it functions in ISM (2.4 GHz) band which is globally freely accessible by everyone,

2. it provides the maximum data rate among the three available modes.

The proposed PHY layer design contains independent Transmitting (Tx) and Receiving (Rx) chains, with each chain consists of a flexible Baseband Processing Unit (BBPU) and a configurable analog part. Each BBPU written in Verilog language is fully parametrized and easily configurable at run time [14, 15]. In [14], the authors have prototyped a IEEE 802.15.4 compliant TRX on SDR platform which is further explored in [15] to achieve zero turnaround time in Time Division Duplex (TDD) operation on SDR front-end. We have further improved the existing architecture of each BBPU by exposing many parameters (so-called radio drivers) from each chain to the upper layers (see control lines in Fig. A.2). The SDR platform used in our setup is composed of Zynq-7000 All Programmable System on Chip (AP SoC) and FMCOMMS2 board [16, 17]. Zynq 7000 AP SoC is further comprised of Programmable Logic (PL) and Processor System (PS). Both BBPUs are implemented in PL part, while FMCOMMS2 board is used as analog RF Front-end.

A.4.2 The Flexible Design for the MAC Layer of IEEE 802.15.4

In addition to the PHY layer, IEEE 802.15.4 standard also specifies the MAC layer that can work in two modes of transmission: non-beacon enabled and beacon-enabled. Non-beacon mode aims to provide high throughput in light traffic scenario, while beacon mode gives deterministic behavior in all types of traffic. To achieve this, non-beacon enabled mode employs un-slotted Carrier Sense Multiple Access/Collision Avoidance (CSMA/CA) MAC protocol, whereas beacon-enabled uses Time Division Multiple Access (TDMA) and slotted CSMA/CA MAC protocols inside so-called super-frame. The details of TDMA and CSMA is outside the scope of this chapter.

To this date, operating systems offer limited possibilities to alter the MAC layer in terms of functionality and timings. For this purpose, the TAISC was developed, which is a cross-platform MAC protocol compiler and execution engine [18]. It divides the MAC layer into two distinct sub-layers: the lower and upper-MAC. The sub-processes running in lower-MAC are mostly hardware-dependent and time-critical (e.g. back-off time execution in CSMA, slot duration in TDMA, etc.). On the other hand, the upper-MAC is hardware-independent and less time-critical (e.g. scheduling of the super-frame in TDMA). TAISC introduces time-annotated radio instructions which the compiler utilizes to generate precise timing information. The execution engine later uses this timing information to schedule instructions. In this way, the engine ensures deterministic behavior of the lower-MAC. Full flexibility is offered by exposing a set of run-time configurable parameters per MAC protocol. It also offers the possibility to interchange (parts of) a MAC protocol by another. TAISC along with a complete network stack is running on ARM processor embedded in PS part of ZYNQ 7000. Since the data rate flexibility is associated

Table A.1: List of some important drivers supported in the proposed TRX.

Driver name	Description
send_radio(buff_tx)	Start packet transmission
rcvd_radio(buff_rx)	Read received packet
set_tx_gain(val)	Change transmitter gain
set_rx_gain()	Change receiver gain
set_channel(ch_no)	Change current RF channel
cca_read()	Perform channel assessment for CSMA
set_data_rate(val)	Change data rate

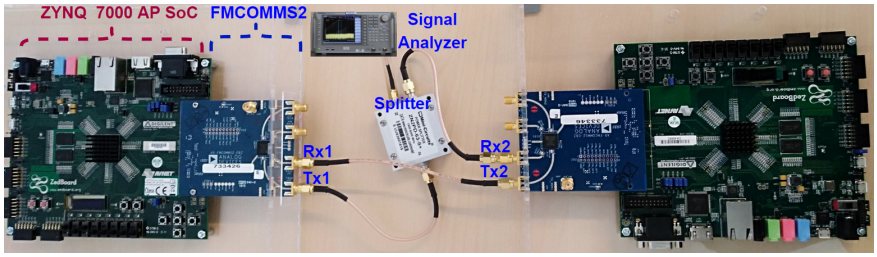


Figure A.3: Experimental setup.

with a PHY layer (e.g., upper and lower limits of a data rate are defined based on PHY layer constraints), therefore we mainly focus on the PHY layer in the later parts of the chapter.

A.5 Results and Discussion

We claim that the proposed PHY layer is flexible, which is achieved by exposing a rich set of radio functions in radio drivers to upper layers. Table A.1 lists some important functionalities supported in the proposed TRX. We focus on introducing a new radio function, namely set_data_rate(), capable of changing the data rate on the fly. When this function is called, the following three items are modified in the background: (1) sampling frequency of DAC/ADC, (2) the processing speed of BBPUs, (3) and the BW of operating channel at analog RF front-end. Since these changes are associated with operating speed and utilize the same hardware therefore the proposed TRX employs the same hardware accelerator to achieve variable date rate.

Data rate directly influences the performance of the proposed TRX. Therefore, we have experimentally measured system performance under different data rates. The performance metrics considered in our experimental setup are latency, coverage, and sensitivity.

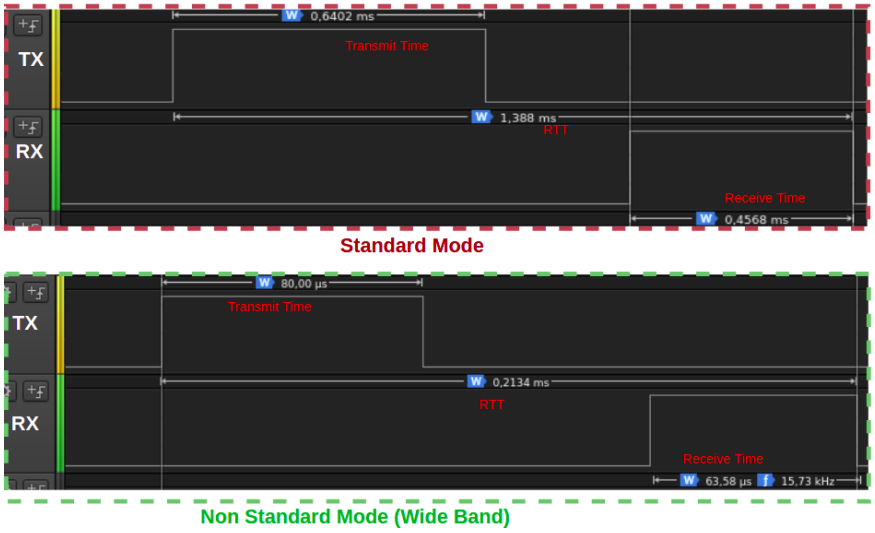


Figure A.4: RTT values of the proposed TRX with Packet size of 20 bytes measured by toggling corresponding GPIO pins and visualized in Seleae logic analyzer.

A.5.1 Experimental Setup

The Fig. A.3 illustrates the setup deployed to evaluate the above-mentioned performance metrics. The setups consist of two TRXs, a Mini-Circuits' ZN2PD2-63C power splitter, SMA cables, and a signal analyzer MS2690A [19, 20].

A.5.2 Latency Measurements

The latency is measured in terms of Round-Trip Time (RTT), the time required by a packet to travel from a source to a destination and back again. Since the proposed TRX can be configured to a standard mode or non-standard mode, RTT is measured in both modes. Logic Pro 16 USB Logic Analyzer [21] is exploited to measure the RTT. The logic analyzer samples the General Purpose Input and Output (GPIO) pins of the proposed TRX with a sampling frequency of 25 Msps, where GPIO pins are toggled upon the start of a packet transmission and at the end of packet reception (see Fig. A.4). The data rate of the TRX running in standard mode is 250 kbps, while the data rate in non-standard mode can be set either higher or lower than 250 kbps depending upon the availability of free spectrum. The most critical hardware component that defines the lower and upper limits of data rate of the proposed TRX is the sampling rate of the RF front-end FMCOMMS2 [17]. The minimum and maximum sampling rates that FMCOMMS2 supports are 520.9 ksps, 61.44 Msps respectively. Fig. A.4 reports the RTT values of the proposed TRX with a packet size of 20 bytes. It can be seen in Fig. A.4 that the TRX

Table A.2: Measured performance of the proposed TRX in terms of coverage range, sensitivity and RTT.

Parameters	Standard Mode	Non Standard Mode	
	Standard BW	Narrowest BW	Widest BW
Data Rate (kbps)	250	31.25	≈2000
Bandwidth (MHz)	2	0.25	≈16
BB Sampling Frequency (MHz)	8	1	61.44
RTT (ms)	1.39	14.08	0.213
Sensitivity (dBm)	-98	-107	-90
Calculated Coverage Range (m)	123	347	49

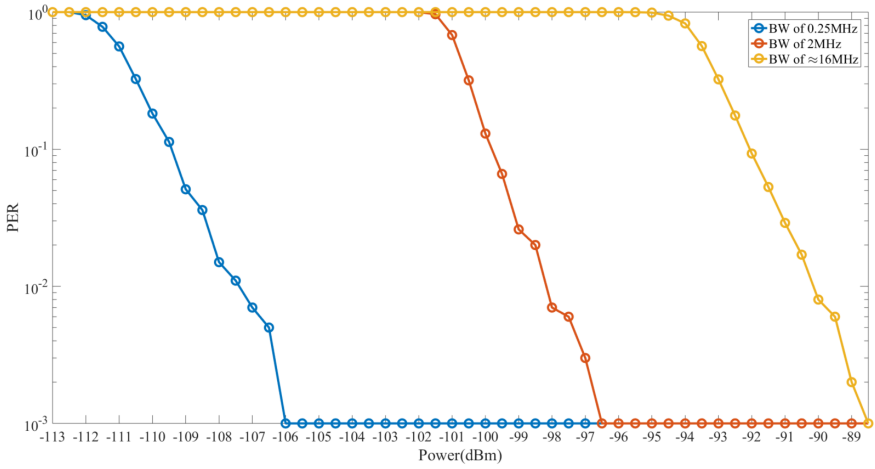


Figure A.5: Receiver sensitivity of the proposed TRX under different settings.

operating in non-standard mode (i.e. 8th time higher data rate) takes almost 8 times as short as standard compliant mode time. It is worth noting that the Rx time (see in Fig. A.4) is always shorter because it does not include preamble 5 bytes.

The maximum and minimum BWs, data rate and their corresponding measured RTT values that the proposed TRX supports are listed in Table A.2. The approximation sign in Table A.2 is used because the maximum Baseband (BB) sampling rate supported by the RF fron-end is 61.44 Msps, our BB is oversampled by a factor of 4 in order to be mapped to the half sine pulse as specified in IEEE 802.15.4, hence the widest BW in non standard mode is 61.44 MHz/4, which is approximately 16 MHz. The oversampling ratio is applicable for all modes, and the BB sampling frequency is also mentioned correspondingly in Table A.2.

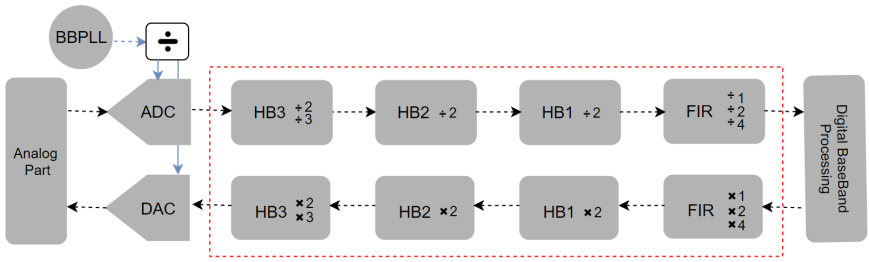


Figure A.6: TX and RX chains of FMCOMMS2.

A.5.3 Sensitivity Measurements

Since the proposed TRX complies with IEEE 802.15.4 standard, the RX sensitivity is measured in accordance with the standard [6]. The RX sensitivity is defined as the minimum input power at which the RX provides Packet Error Rate (PER) of 1%. The setup shown in Fig. A.3 is used to transceive packets, while PER is calculated by employing (A.1),

$$\left(\frac{RX_{packet}}{TX_{packet}}\right) \times 100 \tag{A.1}$$

Note Where TX_{packet} and RX_{packet} represent the numbers of transmitted and successfully decoded packets respectively. To determine the PER, we transmit 1000 packets with each packet contains 20 bytes in the air. This transmission is repeated with different Tx power that is managed by Transmitter (TX)'s attenuator in FMCOMMS2. The linear relation between TX's attenuator (in dB) and RX's absolute power (in dBm) reported in [15] is used to acquire RX's sensitivity. The measured sensitivity values of the proposed TRX with different BWs are presented in Fig. A.5. The RX sensitivity is improved by 9 dB in narrow BW mode and is reduced by 8 dB in wide BW mode when it is benchmarked against the standard mode (the results can be well explained by the theory i.e., the sensitivity of a RX is increased by 3 dB every time when data rate is decreased by 2.).

We have subsequently calculated the coverage range based on new sensitivity values. Texas Instrument (TI) introduces TI RF Range Estimator [22] to estimate a range for sub-1 GHz and 2.4 GHz devices. By keeping Tx power 0 dBm, the estimator is utilized to calculate the ranges of the proposed TRX at different data rate and the calculated values are listed in Table A.2.

A.5.4 Comparison with the Existing State-of-the-art Solutions

Table A.3 represents a comparison of our work with the existing state-of-the-art works. In addition to supporting the run-time data rate configurations, the time required to configure it is also important, because how fast a system responds to a

Table A.3: A Comparison of the proposed solution with the existing state-of-the-art solutions.

Parameters	[13]	CC1352P [8]	Our Work
Data Rate Range	20, 40, 250 kbps	2.5-2000 kbps	31.25-2000 kbps
Best Sensitivity	NA	-122 dBm	-107 dBm
Frequency Range	868 MHz, 902 MHz, 2.4 GHz	2.4 GHz, Sub-1 GHz	70 MHz to 6 GHz
Standard(s) Support	IEEE 802.15.4 PHY and MAC	IEEE 802.15.4 PHY and MAC	IEEE 802.15.4 PHY and MAC
Data Rate Configuration time	NA	Up to 500 μs	Up to 1600 μs

change in an environment depends on latency needed to reconfigure its data rate. In default configuration, the adaptation of data rate of the proposed TRX is accomplished by performing the following two steps: (1) the frequencies of BaseBand Phase Locked Loop (BBPLL), ADC/DAC, Half-Band (HB) filters, and Finite Impulse Response (FIR) filters in FMCOMMS2 are changed, (2) subsequently the calibration of BBPLL is performed (see Fig. A.6). It is observed that this calibration takes up to 421 ms, which makes the proposed TRX less responsive to RF environment change. We keep the clocks of BBPLL, and ADC/DAC fixed, and change the frequencies of HB3, HB2, HB1, and FIR, therefore, the calibration of BBPLL is no more needed. Thanks to this approach, the proposed TRX can change the data rate in only 1.6 ms, making it more responsive to the wireless RF environment changes.

In comparison with [13] in which each data rate corresponds to a dedicated hardware, the proposed TRX utilizes the same hardware for various data rate. In contrast to CC1352P that has flexible data rate feature viable only in Sub-1 GHz band, the proposed TRX is able to configure the data rate in a wider band (i.e., 70 MHz to 6 GHz). In terms of data rate adaptation, CC1352P employs different modulation techniques which definitely consuming extra silicon area, while the proposed TRX has achieved the same results without adding any extra hardware (i.e., by simply changing the sampling frequency).

A.6 Conclusion

In order to comply with standards, the existing state-of-the-art wireless devices for WSNs exhibit fixed data rate (or fixed BW). These devices perform well in a controlled RF environment, but their efficiency deteriorates in a diverse and dynamic RF environment (e.g., in a fully crowded spectrum). Therefore, we introduce an SDR based TRX that can be configured in IEEE 802.15.4 compliant mode and

non-standard modes (i.e., wider BW or Narrower BW) depending on the spectrum occupancy. Flexible feature of an SDR enables the proposed TRX to achieve the run-time data rate adaptation without utilization of any extra hardware. Narrow BW mode of the proposed TRX outperforms the standard mode in terms of reliability in heavily occupied radio spectrum environment, or offers a better coverage range with a given transmit power, whereas wide BW mode may provide relatively low latency communication link and higher throughput when more radio spectrum is available. Subsequently, performance metrics such as latency in term of RTT, coverage, and sensitivity of the proposed TRX are measured under various data rate configurations. Experimental results reveal that the sensitivity of the proposed TRX in standard mode is -98 dBm which is enhanced by 9 dB when it is configured in narrow BW mode and is decreased by 8 dB when it is configured in wide BW mode.

Acknowledgment

This work was supported by the European Union's Horizon 2020 Research and Innovation Programme (ORCA project) under Grant 732174.

References

[1] S. K. Sharma, B. Bhushan, R. Kumar, A. Khamparia, and N. C. Debnath. *Integration of WSNs Into Internet of Things: A Security Perspective*. CRC Press, 2021.

[2] *ZigBee Specification*, 2015. Available from: https://zigbeealliance.org/wp-content/uploads/2019/11/docs-05-3474-21-0csg-zigbee-specification.pdf.

[3] *System Engineering Guidelines IEC 62591 WirelessHART*, 2016. Available from: https://www.emerson.com/documents/automation/engineering-guide-system-engineering-guidelines-iec-62591-wirelesshart_en-79900.pdf.

[4] M. Nixon and T. R. Rock. *A Comparison of WirelessHART and ISA100. 11a*. Whitepaper, Emerson Process Management, pages 1–36, 2012.

[5] *IETF IPv6 over Low power WPAN (6LoWPAN)*, 2012. Available from: https://datatracker.ietf.org/wg/6lowpan/documents/.

[6] *IEEE Standard for Low-Rate Wireless Networks*, 2020. Available from: https://standards.ieee.org/ieee/802.15.4/7029/.

[7] *CC1312R SimpleLink High-Performance Sub-1 GHz Wireless MCU*, 2020. Available from: http://www.ti.com/lit/ds/symlink/cc1312r.pdf.

[8] *CC1352P SimpleLink™ High-Performance Dual-Band Wireless MCU With Integrated Power Amplifier*, 2021. Available from: http://www.ti.com/lit/ds/symlink/cc1352p.pdf.

[9] T. Schmid. *Gnu radio 802.15. 4 en-and decoding*. unpublished document and source, 2006.

[10] *The Universal Software Radio Peripheral (USRP)*, 2022. Available from: https://www.ettus.com/.

[11] *GNU Radio - The Free & Open Source Radio Ecosystem*, 2022. Available from: http://www.ti.com/lit/ds/symlink/cc1352p.pdf.

[12] B. Bloessl, C. Leitner, F. Dressler, and C. Sommer. *A GNU radio-based IEEE 802.15. 4 testbed*. 12. GI/ITG KuVS Fachgespräch Drahtlose Sensornetze (FGSN 2013), pages 37–40, 2013.

[13] A. Massouri and T. Risset. *FPGA-based implementation of multiple PHY layers of IEEE 802.15. 4 targeting SDR platform*. In SDR-WInnComm. Wireless Innovation Forum, 2014.

[14] T. Kazaz, X. Jiao, M. Kulin, and I. Moerman. *WiSCoP-Wireless Sensor Communication Prototyping Platform*. arXiv preprint arXiv:1612.02900, 2016.

[15] M. Aslam, X. Jiao, W. Liu, and I. Moerman. *An approach to achieve zero turnaround time in TDD operation on SDR front-end*. IEEE Access, 6:75461–75470, 2018. doi:https://doi.org/10.1109/ACCESS.2018.2883253.

[16] *An Overview of Zynq-7000 SoC Data Sheet*, 2018. Available from: https://docs.xilinx.com/v/u/en-US/ds190-Zynq-7000-Overview.

[17] *User Guide of AD-FMCOMMS2-EBZ an FMC Board for the AD9361*, 2020. Available from: https://wiki.analog.com/resources/eval/user-guides/ad-fmcomms2-ebz.

[18] B. Jooris, J. Bauwens, P. Ruckebusch, P. De Valck, C. Van Praet, I. Moerman, and E. De Poorter. *Taisc: a cross-platform mac protocol compiler and execution engine*. Computer Networks, 107:315–326, 2016. doi:https://doi.org/10.1016/j.comnet.2016.03.027.

[19] *Data Sheet of Mini-Circuits' ZN2PD2-63+ High-Power Splitter/Combiner*, 2022. Available from: https://www.minicircuits.com/pdfs/ZN2PD2-63+.pdf.

[20] *Operational Manual of MS2690A/MS2691A/MS2692A Signal 699 Analyzer*, 2022. Available from: https://www.anritsu.com/en-US/test-measurement/products/ms2692a.

[21] *Logic Pro 16 USB Logic Analyzer*, 2018. Available from: http://downloads.saleae.com/specs/Logic+Pro+16+Data+Sheet.pdf.

[22] *TI RF Range Estimator*, 2018. Available from: https://www.ti.com/tool/RF-RANGE-ESTIMATOR.

B

The Engineering Work of Wi-Fi Implementation

Chapter 3 introduces a hardware efficient design for IEEE 802.11ax compliant Multi User (MU)-Orthogonal Frequency Division Multiple Access (OFDMA) Transceiver (TRX). Whereas, Chapter 5 presents a hardware efficient design for clock synchronization algorithm across wired-wireless network. These Hardware (HW) efficient designs proposed in Chapters 3 and 5 are validated using $openwifi$. This Appendix first describes the $openwifi$ and then summarizes the contributions added on top of exisitng $openwifi$'s design during this PhD thesis.

⋆⋆⋆

Unlike closed-source (or proprietary) technologies (or projects) which are essentially protected and bound via intellectual property, open-source projects are publicly available and free to use. There are many open-source projects in Software Defined Radio (SDR) community such as srsLTE and Open Air Interface (OAI) which have been widely used for 4G/5G research [1, 2]. Recently, Xianjun et al. [3] have developed the free and open source full stack IEEE 802.11 SDR implementation named $openwifi$. The current implementation of $openwifi$ supports IEEE 802.11a/g/n standards.

As shown in the Fig. B.1, the architecture of $openwifi$ consists of three main hardware components, i.e., Advanced RISC Machines (ARM) processor, Field Programmable Gate Arrays (FPGA) and Radio Frequency (RF) front-end. The ARM processor and FPGA are embedded on the same System on Chip (SoC)

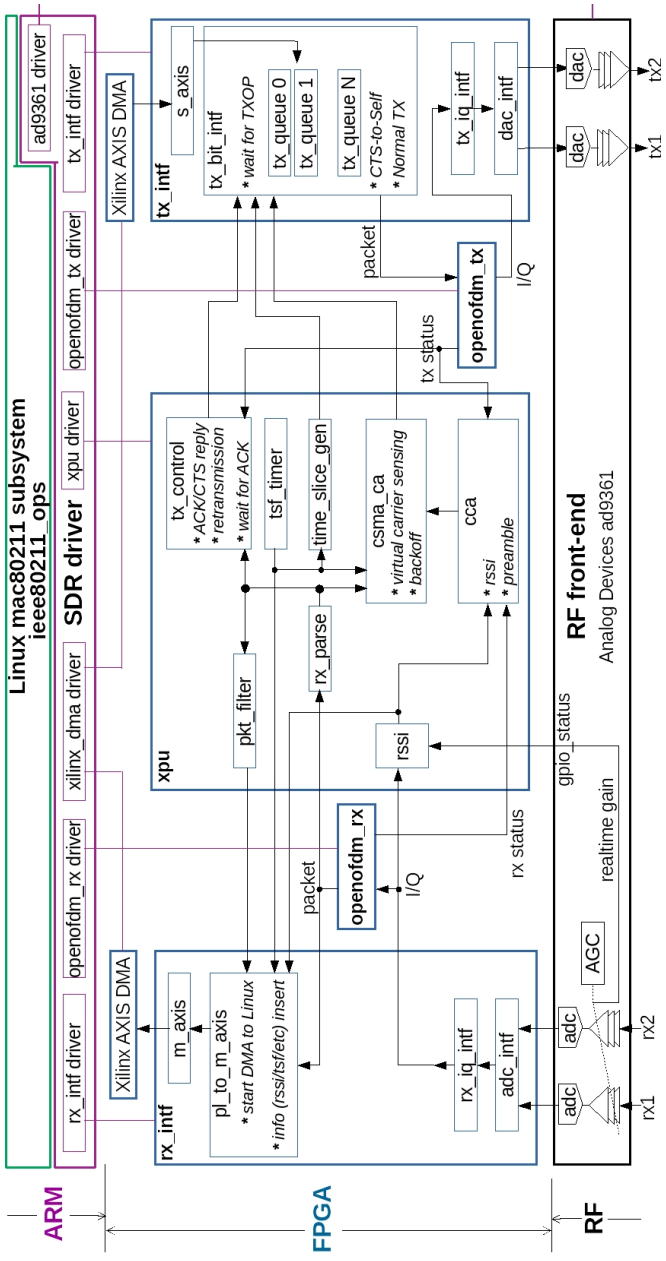


Figure B.1: System architecture of openwifi [4].

(i.e., Zynq SoC [5]). High speed FPGA Mezzanine Card (FMC) interface is used to connect the analog RF front-end (i.e., AD-FMCOMMS2/3/4 [6]) with SoC. In $openwifi$, the application, UDP/TCP IP, upper Media Access Control (MAC) layers, and radio drivers are running on embedded processor. For upper layers and upper MAC, the existing Linux network and IEEE 802.11 stacks are used. The lower MAC and Physical (PHY) layers are implemented on FPGA[1].

In Chapter 3, the PHY layer of $openwifi$ is modified and support of MU-OFDMA is added. Subsequently, the existing network stack of $openwifi$ is leveraged to verify the functionality of MU-OFDMA. In Chapter 5, the tsf_timer submodule of xpu module in PHY layer of $openwifi$ is modified to support skew correction feature which qualifies the Timing Synchronization Function (TSF) clock as a hardware Precision Time Protocol (PTP) clock (see in the Fig. B.1). Later, the radio driver and communication stack of linux are changed to transmit/receive the PTP messages as well as to perform the correction of PTP hardware clock.

Fig. B.2 shows the branching of $openwifi$ in github. The $openwifi$ is maintained in $internal\ master$ and $public\ master$ branches. The $public\ master$ contains the features of $openwifi$ which are publicly accessible and free to use, whereas $internal\ master$ is maintained internally and has more advanced features than $public\ master$. In this PhD dissertation, two branches are created from $internal\ master$ to implemented MU-OFDMA ($Ch.3\ (MU-OFDMA)$ in Fig. B.2) and PTP over wired-wireless network ($Ch.5\ (PTP)$ in Fig. B.2).

[1] more details about hardware architecture of $openwifi$ can be found in [4]).

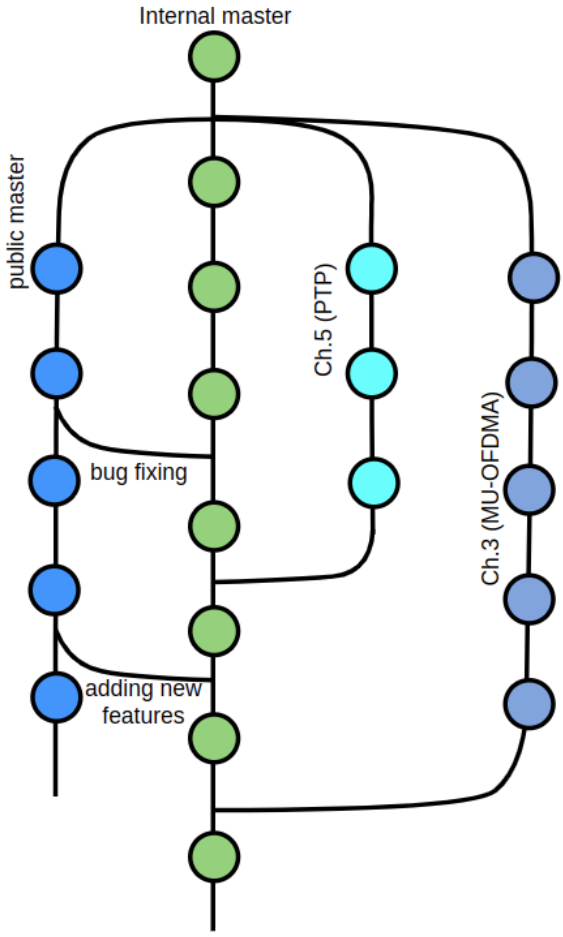


Figure B.2: Branching of openwifi in github.

References

[1] I. Gomez-Miguelez, A. Garcia-Saavedra, P. D. Sutton, P. Serrano, C. Cano, and D. J. Leith. *srsLTE: An open-source platform for LTE evolution and experimentation*. In Proceedings of the Tenth ACM International Workshop on Wireless Network Testbeds, Experimental Evaluation, and Characterization, pages 25–32, 2016.

[2] N. Nikaein, M. K. Marina, S. Manickam, A. Dawson, R. Knopp, and C. Bonnet. *OpenAirInterface: A flexible platform for 5G research*. ACM SIGCOMM Computer Communication Review, 44(5):33–38, 2014.

[3] J. Xianjun, L. Wei, and M. Michael. *open-source IEEE802.11/Wi-Fi baseband chip/FPGA design*, 2021. Available from: https://github.com/open-sdr/openwifi.

[4] X. Jiao, W. Liu, M. Mehari, M. Aslam, and I. Moerman. *openwifi: a free and open-source IEEE802. 11 SDR implementation on SoC*. In 2020 IEEE 91st Vehicular Technology Conference (VTC2020-Spring), pages 1–2. IEEE, 2020.

[5] *An overview of Zynq-7000 SoC Data Sheet*, 2022. Available from: https://www.xilinx.com/support/documentation/data_sheets/ds190-Zynq-7000-Overview.pdf.

[6] *AD-FMCOMMS3-EBZ: An AD9361 Software Defined Radio Board*, 2021. Available from: https://www.analog.com/en/design-center/evaluation-hardware-and-software/evaluation-boards-kits/eval-ad-fmcomms3-ebz.html.