Multicarrier Faster-than-Nyquist Signaling Transceivers: From Theory to Practice

Dasalukunte, Deepak

2011

Multicarrier Faster-than-Nyquist Signaling Transceivers

From Theory to Practice

Deepak Dasalukunte

Lund University

Ph.D Thesis, January 2012

Department of Electrical and Information Technology
Lund University
Box 118, SE-221 00 LUND
SWEDEN

This thesis is set in Computer Modern 10pt
with the LaTeX Documentation System

Series of licentiate and doctoral theses
ISBN 978-91-7473-223-8
ISSN 1654-790X
No. 36

Printed in Sweden by *Tryckeriet i E-huset*, Lund.
December 2011.

Abstract

The demand for spectrum resources in cellular systems worldwide has seen a tremendous escalation in the recent past. The mobile phones of today are capable of being cameras taking pictures and videos, able to browse the Internet, do video calling and much more than an yesteryear computer. Due to the variety and the amount of information that is being transmitted the demand for spectrum resources is continuously increasing. Efficient use of bandwidth resources has hence become a key parameter in the design and realization of wireless communication systems. Faster-than-Nyquist (FTN) signaling is one such technique that achieves bandwidth efficiency by making better use of the available spectrum resources at the expense of higher processing complexity in the transceiver.

This thesis addresses the challenges and design trade offs arising during the hardware realization of Faster-than-Nyquist signaling transceivers. The FTN system has been evaluated for its achievable performance compared to the processing overhead in the transmitter and the receiver. Coexistence with OFDM systems, a more popular multicarrier scheme in existing and upcoming wireless standards, has been considered by designing FTN specific processing blocks as add-ons to the conventional transceiver chain. A multicarrier system capable of operating under both orthogonal and FTN signaling has been developed. The performance of the receiver was evaluated for AWGN and fading channels. The FTN system was able to achieve 2x improvement in bandwidth usage with similar performance as that of an OFDM system. The extra processing in the receiver was in terms of an iterative decoder for the decoding of FTN modulated signals. An efficient hardware architecture for the iterative decoder reusing the FTN specific processing blocks and realize different functionality has been designed. An ASIC implementation of this decoder was implemented in a 65nm CMOS technology and the implemented chip has been successfully verified for its functionality.

To my parents

Preface

This thesis summarizes my research work carried out in the Digital ASIC group at the Department of Electrical and Information Technology for the Doctoral degree (Ph.D) in Circuit Design. The main contributions of the thesis are:

[1] D. Dasalukunte, F. Rusek, and V. Öwall, "Improved memory architecture for multicarrier faster-than-Nyquist iterative decoder," in *Proc. of IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*, Chennai, Jul 2011.

[2] D. Dasalukunte, F. Rusek, and V. Öwall, "Multicarrier faster-than-Nyquist signaling transceivers: Hardware architecture and performance analysis," *IEEE Transactions on Circuits and Systems-I (TCAS-I)*, vol. 58, no. 4, pp. 827-838, Apr 2011.

[3] D. Dasalukunte, F. Rusek, and V. Öwall, "An iterative decoder for multicarrier faster-than-Nyquist signaling systems," in *Proc. of IEEE International Conference on Communications (ICC)*, Cape Town, May 2010.

[4] D. Dasalukunte, K. Ananthanarayanan, M. Kandasamy, F. Rusek, and V. Öwall, "Hardware implementation of mapper for faster-than-Nyquist signaling transmitter," in *Proc. of IEEE NORCHIP*, Trondheim, Nov 2009.

[5] D. Dasalukunte, F. Rusek, J. B. Anderson, and V. Öwall, "A transmitter architecture for faster-than-Nyquist signaling systems," in *Proc. IEEE International Symposium on Circuits and Systems (ISCAS)*, Taipei, May 2009.

[6] S. Mehmood, D. Dasalukunte, and V. Öwall, "Hardware architecture of IOTA pulse shaping filters for multicarrier systems," under first revision in *IEEE Transactions on Circuits and Systems-I* Dec. 2011.

[7] D. Dasalukunte, S. Mehmood, and V. Öwall, "Complexity analysis of IOTA filter architectures in faster-than-Nyquist multicarrier systems," in *Proc. of IEEE NORCHIP*, Lund, Nov 2011.

I have also contributed to the following articles during my stint at the Department of EIT.

[8] D. Noguet, M. Laugeois, X. Popon, P. Balamuralidhar, N. Sortur, M. Lobeira, D. Dasalukunte, Z. Bakirtzoglou and C. Dehos, "An MC-SS platform for short-range communications in the Personal Network context," in *EURASIP Journal on Wireless Communications and Networking*, 2008.

[9] D. Dasalukunte, and V. Öwall, "A generic hardware MAC for wireless personal area network platforms," in *Proc. International Symposium on Wireless Personal Multimedia Communications (WPMC)*, Saariselka, Sep. 2008.

[10] D. Dasalukunte, A. Pålsson, M. Kamuf, P. Persson, R. Veljanovski and V. Öwall, "Architectural optimization for low power in a reconfigurable UMTS filter," in *Proc. International Symposium on Wireless Personal Multimedia Communications (WPMC)*, San Diego, 2006.

Contents

Acknowledgments

I would like to take this opportunity to thank those that have helped me during my years at the Department of Electrical and Information Technology as a PhD student.

First of all, my sincere gratitude and thanks to my supervisor Prof. Viktor Öwall for taking me as his PhD student, when I myself was not confident that I could come this far. Thanks for providing such a wonderful opportunity and for all your support and encouragement.

Thanks to my co-supervisor Dr. Fredrik Rusek who has been very helpful and patient in teaching me basics and advanced topics in Communication theory, I must admit, I still lack some. Also, special thanks to Prof. John B Anderson.

I have enjoyed my work and research in the Digital ASIC group and the Dept. of EIT. I would like to thank Johan, my contemporary, and current colleagues Joachim, Isael, Chenxin, Yasser, Reza, Oskar, Liang, Hemanth and Rakesh for all the constructive and interesting discussions I had in the group. It was a pleasure working with you all. Also, thanks to my former colleagues for their guidance when I started my PhD. I also thank my other colleagues and friends at the department for interesting coffee and lunch time discussions.

I would like to thank Prof. Borivoje Nikolic for hosting me at the Berkeley Wireless Research Center, UC Berkeley during Jan-Apr 2010. Thanks to Antonio Vicent from ST-Ericsson for his expert help during my chip tapeout.

My profound thanks to Pia for fixing almost everything on the administrative side for which I approached her, thanks to Erik and Stefan for support with the computers and CAD tools. Many thanks to Lars and rest of the administrative staff for getting things done and making sure things are in place so that I could focus better on my work.

Finally, I would like to express my utmost gratitude to my family. Thank you amma, daddy and Anu for your unconditional support and sacrifices that has made me what I am today!

Deepak Dasalukunte

Acronyms

3GPP 3rd Generation Partnership Project.

ASIC Application Specific Integrated Circuit.

AWGN Additive White Gaussian Noise.

BCJR Bahl-Cocke-Jelenik-Raviv.

BER Bit Error Rate.

CMOS Complementary Metal Oxide Semiconductor.

DEMUX De-Multiplexer

DFT Discrete Fourier Transform.

dp-RAM Dual Port RAM.

DVB Digital Video Broadcasting.

FFT Fast Fourier Transform.

FPGA Field Programmable Gate Array.

FTN Faster-than-Nyquist.

ICI InterCarrier Interference.

IID Independent, Identically Distributed.

IOTA Isotropic Orthogonal Transform Algorithm.

ISI InterSymbol Interference.

ITRS International Technology Roadmap for Semiconductors.

LLR Log-Likelihood Ratio.

LR Likelihood Ratio.

LTE Long Term Evolution.

LUT Look-Up Table.

MAP Maximum APosteriori.

MLM Max-Log-MAP.

MCM Multi-Carrier Modulation.

MF Matched Filter.

MUX Multiplexer.

OFDM Orthogonal Frequency Division Multiplex.

OQAM Offset-Quadrature Amplitude Modulation.

OQPSK Offset-Quadrature Phase Shift Keying.

PAPR Peak-to-Average Power Ratio.

RAM Random Access Memory.

RTL Register Transfer Logic.

SNR Signal-to-Noise Ratio.

SIC Successive Interference Cancellation.

sp-RAM Single Port RAM.

VHDL VHSIC Hardware Description Language.

VHSIC Very High Speed Integrated Circuit.

VLSI Very Large Scale Integration.

WLAN Wireless Local Area Network.

Symbols

k Sub-carrier index of FTN symbols.

ℓ Time index of FTN symbols.

m Sub-carrier index of orthogonal symbols.

n Time index of orthogonal symbols.

N Number of sub-carriers in an orthogonal multicarrier system.

M Number of time instances in a orthogonal multicarrier system.

K Block size of transmitted/received information ($= N \times M$).

N_{FTN} Number of sub-carriers in a multicarrier FTN system.

M_{FTN} Number of time instances in a multicarrier FTN system.

i Imaginary number ($= \sqrt{-1}$).

N_t Number of projection points along time.

N_f Number of projection points sub-carriers.

T Time period of a rectangular pulse carrying information in an orthogonal/OFDM system.

L_{ext} Extrinsic log-likelihood ratio.

T_Δ Time spacing between two adjacent FTN symbols.

F_Δ Frequency/sub-carrier spacing in the FTN system.

T_{rep} Repetition rate in time.

F_{rep} Repetition rate in frequency.

$\check{x}_{k,\ell}$ Interference canceled symbols

$\bar{x}_{k,\ell}$ Reconstructed FTN symbols are the receiver.

$\hat{x}_{k,\ell}$ Soft symbol + Interference.

$x'_{m,n}$ Projected FTN symbol on the orthogonal basis function at sub-carrier m and time instance n.

$x_{k,\ell}$ FTN information symbol.

$\tilde{x}_{k,\ell}$ Soft symbol.

Chapter 1

Introduction

This thesis deals with inter disciplinary work in wireless communication and VLSI, addressing challenges that arise when realizing wireless communication algorithms in hardware. The algorithms under study are those associated with improving the efficiency of the resources used for information transmission in a wireless system. One primary resource in the wireless system is the frequency band of operation. The frequencies are generally allocated by a governmental/regulatory bodies such as the Body of European Regulator for Electronic Communications (BEREC) [BER], Federal Communications Commission (FCC) [FCC], Telecom Regulatory Authority of India (TRAI) [TRA] to name a few. In this context, a wireless system primarily refers to a mobile phone or a handheld device capable of communicating wirelessly to another device or a base station.

Since its invention, the mobile phone has experienced explosive growth evolving from being a mere voice communicating device to more than a computer in the recent past. Ever since, there has been a rising demand for information transmission and reception. Though the amount of bandwidth available has also increased, the demand has infused severe competition amongst the mobile operators who are paying a very high premium to own spectrum resources. The recent 3G and broadband wireless spectrum auction in India in 2010 fetched USD 23.5 billion for the 5MHz nationwide spectrum in 20 circles [Wik]. In Spain, the 4G mobile licenses is expected to bring EUR 1.5 billion from the bidding of a total of 310MHz in different frequency bands [LG11]. At the same time, advances in semiconductor technology are pushing the limits of processing by cramming in more and more electronics into microchips. This means that the small sized chips can now accommodate more complex circuits. The main goal of this thesis is to realize architectures that are moderately more

complicated than existing ones and efficiently use the expensive bandwidth resources.

In the following sections an extended introduction to the two research topics, wireless communications and VLSI, are discussed. Section 1.1 provides the motivation and a brief background to FTN signaling. Section 1.2 briefs state-of-the-art research carried out in parallel to that presented in this thesis, also intended for improving bandwidth efficiency. Section 1.3 discusses various approaches/technologies available for hardware implementation. Depending on the application, different approaches prove to be suitable for different applications. Here, the focus is mostly on realizing the wireless algorithms in hardware and the choices are made on this basis.

1.1 Motivation for FTN signaling

Wireless communication dates back to late 1800's with pioneering contributions from G. Marconi, R. Fessenden, J.C. Bose, N. Tesla and many others. Since then it has come a long way in different forms as telegraphy, radio broadcasting, television, and in the most recent decades as mobile telephony. The mobile phone was primarily meant to be used for voice based communications, a portable and handheld version of the fixed telephone. The mobile telephony system later evolved into the second generation now popularly known as Global System for Mobile communication (GSM) [GSM]. Since then, the mobile phone has evolved rapidly with non-voice services overtaking the voice based communications. As a result, the mobile phone evolved into more just than a voice communicating device increasing the demand for broadband communication in order to cater for the next generation wireless technologies.

OFDM which stands for orthogonal frequency-division multiplexing is a multi-carrier scheme which is now the heart of several existing and upcoming broadband wireless access technologies such as LTE, WLAN, DVB, IMT-advanced. The sub-carriers carrying data can be seen as a channel transmitting information [Hir81, Cha66, Sal67] and they can vary from a few tens to thousands of sub-carriers. OFDM proved to be a very attractive solution due to its robustness and ease of decoding the modulated signals in a wireless environment. With more and more demand for broadband services such as the internet, video calls, and television, OFDM is being rapidly scaled to meet the demands for the upcoming standards. As a result, there is ever increasing need for improving the existing technologies to cope with the such rapid progress or envision newer schemes that scale with the demand. Though scaling the bandwidth of operation has been possible to a certain extent, it poses challenges in receiver processing and restrictions in terms of frequency usage as it

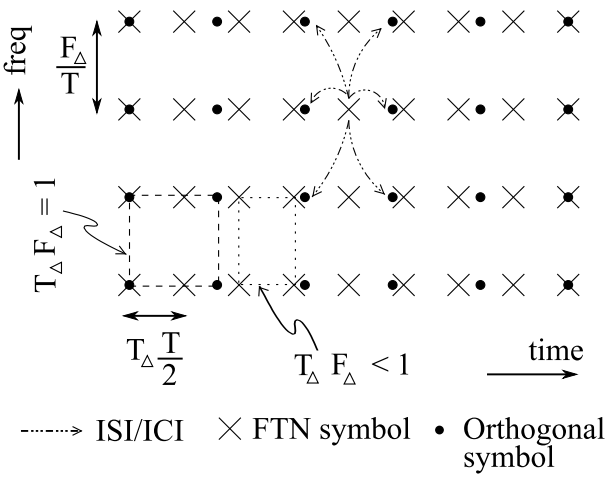

Figure 1.1: FTN and OFDM symbols on the time frequency grid.

is restricted by the regulatory bodies.

A more favorable solution is to make better use of the same available bandwidth in the existing multi-carrier systems. Faster-than-Nyquist (FTN) signaling is one such approach that trades bandwidth for processing complexity. Today bandwidth is a premium resource compared to the processing power that is at disposal in the state-of-the-art silicon technology. Therefore, methods like FTN that trade processing complexity for bandwidth efficiency needs to be investigated. FTN signaling has been theoretically proven to have advantages in single carrier system [RA09a]. However, since most modern wireless systems being multicarrier (mostly OFDM) the impact of FTN signaling in multicarrier environment needs to be evaluated. This especially refers to the algorithm-hardware tradeoff which is the main focus of this research project.

1.1.1 FTN signaling: Background

The concept of FTN was first proposed by J.E. Mazo [Maz75] back in 1975. However, the transceiver involved in such a signaling scheme has a requirement of manifold processing complexity. This is similar to the trend of LDPC codes [Gal63] which were not used until lately due to their inherent implementation complexity. FTN is achieved by signaling information beyond Nyquist's criterion for transmission of symbols without any inter-symbol interference (ISI) [Nyq28]. In other words, to receive the symbols without any ISI, the information can be signaled no faster than the Nyquist rate [PS08, Rus07]. This

is also usually referred to as orthogonal signaling, as there is no interference amongst the symbols.

FTN signaling, on the other hand, packs more symbols than in a conventional orthogonal signaling scheme thus introducing inter-symbol and inter-carrier interference (ISI and ICI). Figure 1.1 graphically shows how the information symbols appear on the time-frequency grid when signaled orthogonally or using FTN signaling. The •s correspond to orthogonal symbols and are separated in time and frequency satisfying Nyquist's condition, hence do not interfere with each other in time or frequency. This system is referred to as an orthogonal system, OFDM being one classic example. The symbols denoted as ×s, are signaled beyond Nyquist's criterion for ISI free transmission resulting in being able to transmit many more symbols than an orthogonal system. The effective improvement in bandwidth usage over the orthogonal system is dependent on the overall compression.

1.2 Prior work and state-of-the-art

Though FTN was developed in the 70's [Maz75], there had not been extensive work either with developing algorithms for FTN or realization on devices for practical usage until recently. A plausible reason is the complexity involved in the realization of such systems, a trend similar to that of the LDPC codes [Gal63]. In the past decade, FTN has received a lot more attention in the research communities worldwide. FTN has also been referred to by other names such as *Spectrally Efficient Frequency Division Multiplexing* (SEFDM), *bandwidth efficiency improvement...* etc. While most are theoretical studies developing algorithms or demonstrating the advantages of using FTN, only very recently there has been some architectural exploration other than the work presented in this thesis. The works listed below highlight what has been researched in each of the articles.

- In [LG03], the preservation of the minimum Euclidean distance for binary signaling in spite of small increase in signaling rate beyond Nyquist is extended to a family of raised-cosine pulses. Practical ways of achieving these gains through iterative joint equalization and decoding scheme is presented.
- [BFC09] investigates the spectral efficiency achievable by a low-complexity symbol-by-symbol receiver. It is shown that when using finite-order constellations, giving up the orthogonality condition can considerably improve the performance.

- [MS10] evaluates FTN transmission and reception in discrete time. A digital communication system with QPSK modulation and AWGN channel is assumed. It is shown that the symbol rate can be increased by 25% without significantly increasing the bit error rate and without requiring increased transmission bandwidth.
- Aspects of fading, Doppler and the robustness of FTN in dispersive channels is discussed in [HZ09].
- [HT04] defines bandwidth efficiency improvement as High Compaction Multi-Carrier Modulation (HC-MCM). It discusses DFT based transmitter and receiver, spectral efficiency and BER performance of the system.
- [YC10] discusses asymptotically achievable information rate of binary FTN signaling.
- [KB10] presents pre-coding of FTN signals and ways of preventing spectrum broadening due to pre-coding.

Under the alternative name of *Spectrally Efficient FDM* (SEFDM) the following works appear in literature:

- [KCRD09] discusses improving bandwidth efficiency by intentionally violating carrier orthogonality in a frequency division multiplexed system and at the expense of receiver complexity. It is shown that it is possible to detect optimally and efficiently FDM signals, with 25% bandwidth gain with respect to analogous OFDM signals.
- [ID11a] is about SEFDM receivers with relatively complex detectors to extract the signal from the ICI created by the loss of orthogonality. Fixed complexity Sphere Decoder (FSD), a hardware friendly approach for the detection of SEFDM signal is proposed.
- SEFDM is extended to time varying channels in [CKRD10]. However, 2 fixed channel realizations are considered in order to evaluate the performance.
- A hardware architecture to realize transmission of SEFDM signals is proposed in [WPID11] and the implementation in [PD11]. The realization is carried out using multiple IFFTs. This is also independently discussed in Section 2.2 together with the drawback in employing such an approach.
- [ID11b] discusses the peak-to-average power ratio (PAPR) reduction. Apart from investigating the standard PAPR reduction techniques, a novel PAPR reduction algorithm termed the Sliding Window PAPR reduction technique is proposed.

FTN related work [Rus07, RA09b] carried out within our department has been the motivation for hardware feasibility studies and also the foundation for the architectures proposed in this work. To the best of author's knowledge, no attempt of hardware implementation for decoding of FTN modulated signals has been made in the literature. An exception is the transmitter architecture in the most recently published work in [WPID11]. The proposed transmitter and receiver architectures in this work remains close to conventional architectures that exist in the standards. This is done in an attempt for seamless switching between an FTN and orthogonal signaling taking advantage of FTN signaling during good channel conditions. The approach in this work has resulted in simplified implementation of FTN signaling and its inclusion into existing systems by simply introducing FTN specific add-on processing blocks. We present an architecture and a successful silicon implementation of the FTN iterative decoder using a state-of-the-art CMOS ASIC process, also a first in the field. Faster-than-Nyquist signaling, having different nomenclatures, have started to gain popularity in the research community mainly because of increasing constraints in the spectrum and relaxing requirements in hardware processing complexity.

1.3 Hardware implementation

The Nobel prize winning invention of the *Integrated Circuit* (in the 1950's) revolutionized the electronics industry. It led to the realization of transistors, a fundamental building block of digital and analog circuits, in a highly compact manner. As a result electronic components could be manufactured on a small piece of silicon die, compared to those made from bulky vacuum tubes, thus resulting in smaller computers and other electronic devices. Over the last 50 years, the number of transistors that could be placed on a silicon die has risen exponentially, from a few to several thousands and today, billions. This trend came to be known as the Moore's law, named after the discoverer who observed and proposed in his paper [Moo65] that the number of transistors on a chip would double every year. The law has continued to be valid until very recently except that the transistor doubling happening every two years instead [ITR]. The International Technology Roadmap for Semiconductors (ITRS) comprises of experts from the semiconductor industry who periodically forecast the technology trends in silicon manufacturing related to microprocessors, memories etc. Figure 1.2 shows the trend from the data extracted from one such report released in 2009 with projections until the year 2025. The doubling of transistors will follow Moore until 2014, after which the trend slows down with the transistor doubling happening every 3 years, according to the ITRS forecast.

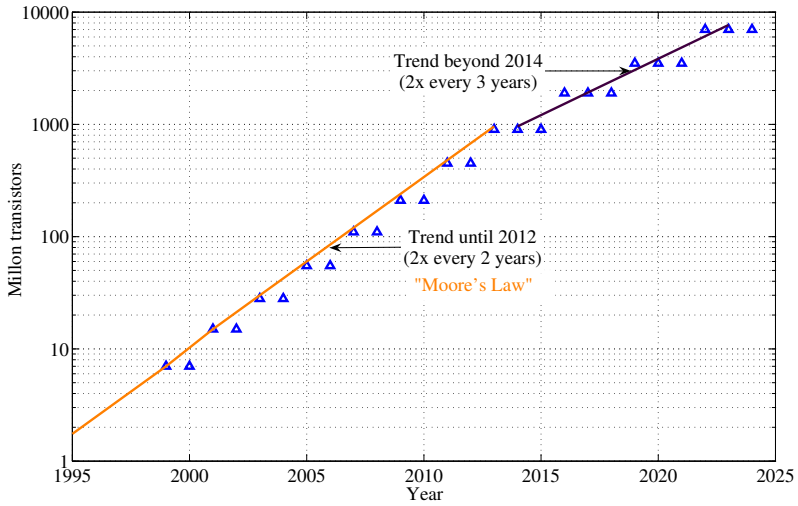

Figure 1.2: Semiconductor technology roadmap from ITRS [ITR].

Figure 1.3 shows Moore's law in action, with data from specific chips from different manufacturers; from the first commercial microprocessor 4004 introduced by Intel [Int] to its latest (on the graph) the 10-core Xeon Westmere-EX.

As the semiconductor industry evolved, the silicon chips were designed to cater to a variety of applications. They could be broadly classified as general purpose processors (CPUs), Digital Signal Processors (DSPs), Field Programmable Gate Arrays (FPGAs) and Application Specific Integrated Circuits (ASICs). In principle, all the above solutions can be used to realize different target applications. However, certain constraints restrict their usage. For instance, applications that run on handheld devices are constrained for power as they run on batteries while desktop computers are more targeted for performance and does not have to be highly limited on the power. Certain other applications might be required to be capable of running different applications at different times bringing up the need for flexibility. Hence these solutions can be broadly classified on the basis of power consumption and flexibility of realizing different applications as they are considered as two important metrics when it comes to realization using hardware.

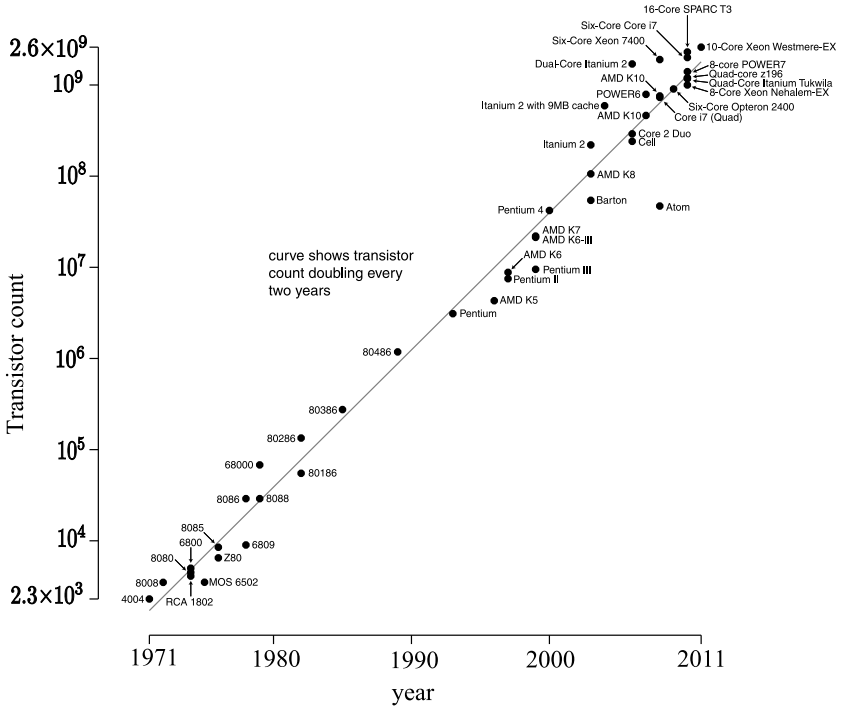

Figure 1.3: Moore's Law as a function of transistor count over the period of 1971-2011 (*Source:"Moore's law" Wikipedia*).

Figure 1.4 shows a general graph of how CPUs, DSPs, FPGAs, and ASICs fare in the flexibility vs. power consumption design space. It is to be noted that this classification has become increasingly obscure with advances in the semiconductor process technology and better tools that help in achieving low power designs. However, the argument of power and flexibility is still applicable in a general context. CPUs are highly programmable and are suitable for general purpose applications and hence are dominantly used in desktop computers, for example, the Intel core i5 processor. DSPs have specialized instructions that can speed up signal processing operations, hence suitable for realizing any signal processing algorithm and TMS320 is one such example of a family of DSPs from Texas Instruments [TI]. FPGAs have programmable logic and can be used to realize custom logic on the fly, but they are known to suffer from poor power performance. The Virtex™ series from Xilinx is a popular example for FPGAs. Finally, ASICs are custom integrated circuits that are

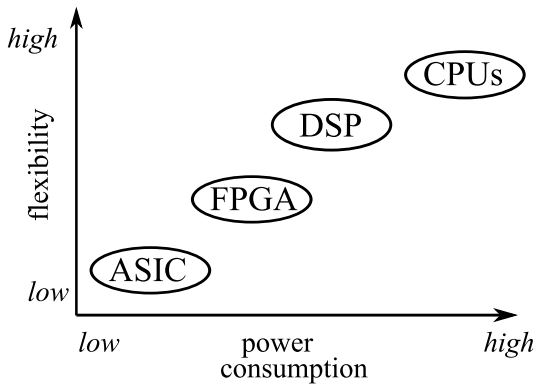

Figure 1.4: Flexibility vs. power consumption of various forms of digital integrated circuits.

fabricated for a particular application. Once manufactured the ASICs cannot be modified to have a different functionality altogether. However, in terms of power consumption, ASICs can be designed to have the best efficiency. These different possibilities discussed so far to realize an algorithm is to motivate the choice in the context of realizing FTN signaling using these solutions.

1.3.1 Algorithm-hardware tradeoffs

Newer algorithms are devised to improve various aspects in wireless communications, for example, error performance and hence improve the overall efficiency of the system. However, for these algorithms to be applied in practice, they have to be architected and implemented on some device, be it a general purpose computer or a specialized chip. Thus, the algorithms are limited by the target platform on which it is implemented and hence have constraints such as:

- Finite wordlengths for number representation.
- Logic and memory defining the area and power consumption.
- Processing capability limitations on how extensively the algorithm can be run.
- Design specification constraints in order to meet speed and throughput rates.

In a wireless system, processing of signals received over a wireless channel involves signal detection, synchronization, demodulation, and decoding. All

of these happens in a layer referred to as the physical layer (PHY), so named because it deals with the physical link connecting the devices (nodes in general) in the network. By 'link' in a wireless system we mean the propagation channel. FTN signaling being discussed in this work is also carried out in this layer of abstraction. The PHY layer has conventionally been implemented on custom hardware due to high demands for power and throughput. Since the hardware is application specific, the integrated circuits developed for this purpose are aptly called ASICs. Though there is also research carried out in this field, generally referred to as Software Defined Radio (SDR), most implementations still largely follows the conventional approach. Software defined radio in the FTN context is an interesting work that still needs to be researched but is not the focus of this work. The main aim of this work is to evaluate the actual overhead in processing one has to pay when adopting FTN to efficiently utilize the bandwidth resources. The CPUs, DSPs and FPGAs have their own overhead in terms of extra logic and other resource based overhead.

For example, with CPUs it is harder to evaluate the exact processing overhead as the processor is usually running several applications together with the one that is of interest to us. In case of FPGAs, the logic overhead cannot be determined accurately as the signal routing is also done using logic resources that are used for realizing the functionality at hand. An ASIC implementation provides an accurate overhead in terms of logic and power consumption as the logic purely corresponds to the functional blocks implemented and hence well motivated.

1.4 Thesis contributions

The contributions in the thesis (hereby referred to as work) is broadly classified as theory/algorithm design sections in Chapters 2 and 3. The hardware architecture, implementation and measurements of the chip is discussed in Chapters 4, 5 and 6. Chapter 7 discusses the hardware architecture and implementation of the IOTA pulse shaping filter.

On the theoretical front, FTN signaling can be improved in several different ways with algorithms that achieve performance close to the theoretic bounds. In this work, major focus is on the feasibility of FTN signaling for its use in practical applications, performance versus hardware complexity tradeoffs and achievable gains when actually implemented in hardware.

The following sub-sections highlight the chapter wise contributions under the two broad categories of theory and implementation.

Theory: FTN signaling in AWGN and fading channels

Chapter 2 (*FTN theory*) discusses the choice of orthogonal basis to be used for multicarrier modulation so as to reduce complexity in the transmitter. It also discusses transmission methods alternate to that proposed and its disadvantages. It has been shown that, operating at sub-optimal points for a particular time-frequency spacing will result in reduced storage complexity. The performance of the receiver for an AWGN channel and the actual gains achieved from the FTN system is elaborated.

In Chapter 3 (*FTN signaling in fading channels*), the receiver processing is extended to combat fading. The modifications carried out to account for fading and steps taken to exploit the fading channel by adaptively using FTN signaling is explained. The achievable data rates while using FTN signaling in a fading environment is presented.

The contents of the chapters are based on the following articles:

1. D. Dasalukunte, F. Rusek, J. B. Anderson, and V. Öwall, "A Transmitter Architecture for Faster-than-Nyquist Signaling Systems," in *Proc.IEEE International Symposium on Circuits and Systems*, Taipei, May 2009.

2. D. Dasalukunte, F. Rusek, and V. Öwall, "An Iterative Decoder for Multicarrier Faster-than-Nyquist Signaling Systems," in *Proc. of IEEE International Conference on Communications*, Cape Town, May 2010.

FTN transmitter, receiver: Hardware architecture, implementation and chip measurements

Chapter 4 (*FTN Transmitter: Hardware architecture and implementation*) details the hardware architecture and the implementation of the mapper used in the transmitter to realize FTN signaling. It also draws a comparison between the complexity in the mapper with respect to the IFFT block, generally considered as a significantly complex block in the transmitter.

Chapter 5 (*FTN Receiver: Hardware architecture and implementation*) elaborates the architecture employed in the FTN decoder and also the re-use of processing blocks to realize different functionalities. Implementation results are provided for both FPGA and ASIC; a comparison in terms of complexity is made between the inner decoder (specific to FTN) and the outer decoder (which is a standard channel decoding approach). Memory optimizations to reduce silicon area and power is also carried out. The measurements from the FTN decoder chip implemented in ST 65nm CMOS is presented in Chapter 6 (*FTN decoder chip:Measurements and results*). The IOTA filter, hardware

architecture and its implementation is presented in Chapter 7 (*IOTA pulse shaping filter in FTN multi-carrier systems*). Though the IOTA pulse shaping is not directly related to the FTN signaling or its decoding, it has influenced in reducing hardware complexity and hence presented as a part of the thesis.

The contents of the chapters are based on the following articles:

1. D. Dasalukunte, F. Rusek, J. B. Anderson, and V. Öwall, "A Transmitter Architecture for Faster-than-Nyquist Signaling Systems," in *Proc.IEEE International Symposium on Circuits and Systems*, Taipei, May 2009.

2. D. Dasalukunte, K. Ananthanarayanan, M. Kandasamy, F. Rusek, and V. Öwall, "Hardware Implementation of Mapper for Faster-than-Nyquist Signaling Transmitter," in *Proc. of IEEE NORCHIP*, Trondheim, Nov 2009.

3. D. Dasalukunte, F. Rusek, and V. Öwall, "Multicarrier Faster-than-Nyquist Signaling Transceivers: Hardware Architecture and Performance Analysis," *IEEE Transactions on Circuits and Systems-I*, vol. 58, no. 4, pp. 827-838, Apr 2011.

4. D. Dasalukunte, F. Rusek, and V. Öwall, "Improved Memory Architecture for Multicarrier Faster-than-Nyquist Iterative Decoder," in *Proc. of IEEE Computer Society Annual Symposium on VLSI*, Chennai, Jul 2011.

5. S. Mehmood, D. Dasalukunte, and V. Öwall, "Hardware architecture of IOTA pulse shaping filters for multicarrier systems," under first revision in *IEEE Transactions on Circuits and Systems-I* Dec. 2011.

Chapter 2

FTN theory

A signaling system is said to be faster-than-Nyquist if the pulses appear at a rate beyond the allowed Nyquist condition for ISI free transmission [Maz75]. Initially, this applied to a single carrier system with pulses overlapping with each other in time. Later this was extended for a multicarrier system (OFDM) [RA09b, Rus07] and in this case the pulses may violate the least required spacing in both time and frequency. As a result there is induced interference in both time and frequency, generally referred to as intersymbol (ISI) and intercarrier interference (ICI). Figure 2.1 shows the way orthogonal and multicarrier FTN symbols appear in the time-frequency lattice where the FTN system has symbols appearing more frequently in time. Though the FTN symbols can be signaled beyond the orthogonality limit in both time and frequency, for illustration the compression is shown only on the time axis. In any case the FTN symbols contribute both to ISI and ICI; the ISI/ICI is conceptually shown in Figure 2.1 for a single FTN symbol.

2.1 Transmission scheme

The information symbols are assumed to be independent and identically distributed (IID) with unit power and the power spectral density of the Additive White Gaussian Noise (AWGN) process is $N_0/2$. The modulation type chosen is offset-Quadrature Amplitude Modulation (OQAM) generally referred to as OFDM/OQAM in literature [Cha66, Sal67, BD74]. OFDM/OQAM allows the use of well localized pulses as pulse shaping filters resulting in transmission at high data rates through wireless channels [Bol03], while in the conventional method of OFDM/QAM it is not possible [LFAB95]. The use of pulse shaping

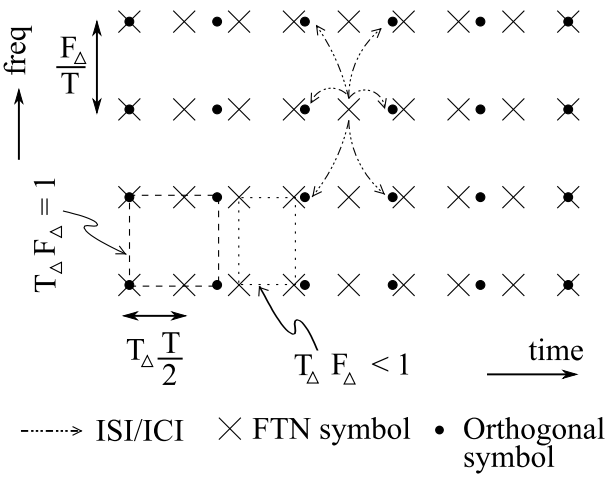

Figure 2.1: Illustration of FTN and Orthogonal symbols on the time-frequency axis.

filters in OFDM/OQAM also has advantages of reduced out-of-band emission and is more robust to carrier frequency offsets [Bol03, RH97]. An OQAM multicarrier modulated signal can be represented as

$$s(t) = \sum_{\ell=-\infty}^{\infty} \sum_{k=0}^{N-1} i^{k+\ell}\, x_{k,\ell}\;\; p\left(t - \ell\frac{T}{2}\right) e^{i\frac{2\pi}{T}kt}, \tag{2.1}$$

where $x_{k,\ell}$ refers to the real valued data symbols that are phase offset with the term $i^{k+\ell}$ and varies depending on the sub-carrier k and time instance ℓ. $p(t)$ is generally a rectangular pulse with time period T in the case of conventional OFDM. In this work though only offset-QPSK is assumed, the results can be extended to higher order modulations. With offset-QPSK, the data symbols take the values of ± 1 which is not the case when higher order modulations are used.

In this work, the FTN system assumes data to be transmitted using Gaussian pulses $g(t)$ as they have excellent time-frequency localization properties. The time period of the Gaussian pulse carrying an information symbol is assumed to be $3T$ for practical reasons, though in theory the pulse has infinite time support. In an FTN multicarrier system that uses Gaussian pulses for information transmission and OQAM modulation, the transmitted signal is

written as

$$s(t) = \sum_{\ell=-\infty}^{\infty} \sum_{k=0}^{N-1} i^{k+\ell}\ x_{k,\ell}\ g\left(t - \ell T_\Delta \frac{T}{2}\right)\ e^{i2\pi \frac{F_\Delta}{T} kt}, \tag{2.2}$$

where k, ℓ refer to the sub-carrier and time indices respectively; $T_\Delta \frac{T}{2}$ is the symbol period between two real valued data symbols ($x_{k,\ell}$) and $\frac{F_\Delta}{T}$ is the sub-carrier spacing. In an orthogonal system with OQAM modulation the time-frequency product would be

$$\left(T_\Delta \frac{T}{2}\right)\left(\frac{F_\Delta}{T}\right) = \frac{T_\Delta F_\Delta}{2}. \tag{2.3}$$

The modulated symbols are transmitted separately as real and complex parts, but at twice the rate of the complex valued symbols. The time frequency product of an orthogonal multicarrier modulation (MCM) with OQAM is $\frac{1}{2}$. That is, while $T_\Delta F_\Delta = 1$ refers to an orthogonal system, $T_\Delta F_\Delta < 1$ refers to a multicarrier system with FTN signaling. As a result, T_Δ and F_Δ can be viewed as compression factors in time and frequency respectively. We point out that although the product $T_\Delta F_\Delta$ specifies the total time-bandwidth consumption of the system, the actual values of T_Δ and F_Δ are important and will be shown in later section. Henceforth, we refer to conventional data symbols being signaled at Nyquist's criterion for ISI free transmission as *orthogonal symbols* and those beyond Nyquist as *FTN symbols*.

There can be several approaches to realize transmission of FTN modulated symbols. One approach is by simply implementing Eqn. (2.2) as is. However, this is not an attractive option as it requires something similar to a Discrete Fourier Transform, but with fractional spacings. Efficient implementations for multicarrier modulations already exist in the form of IFFT, as used in OFDM-based systems. Hence an effort is made to retain this attractive option. However, using only the IFFT introduces complexity in a different dimension and the following section describe the impact of using rectangular windowed sinusoidal basis (basis functions for IFFT) on the overhead complexity. A feasible alternative solution is proposed and evaluated.

2.1.1 Choice of orthogonal basis

In order to use IFFT for multicarrier modulation the Gaussian pulses are to be represented in an orthonormal set of basis functions. Each FTN symbol is represented on the basis functions spanning both time and frequency. The number of basis functions required in time is referred to as N_t and those in

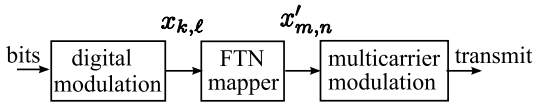

Figure 2.2: A general block diagram of the FTN transmitter using mapper.

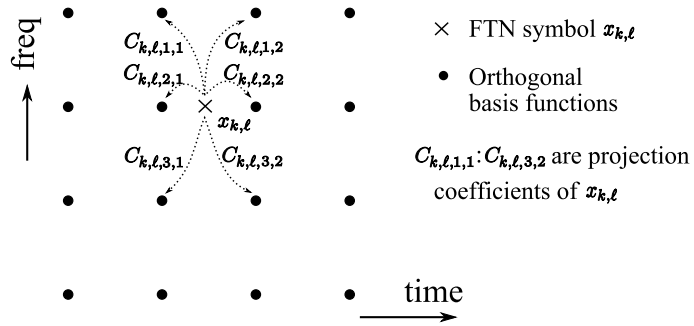

Figure 2.3: Illustration of FTN symbol projected on to an orthogonal basis.

frequency as N_f. Choosing N_t basis functions in time and N_f basis functions in frequency will require $\mathcal{O}(N_t \times N_f)$ computations for each FTN symbol. Let $\psi(t)$ be the basis pulse forming the orthonormal basis $\{\psi_{m,n}(t)\}$ defined as:

$$\psi_{m,n}(t) \triangleq i^{m+n}\ \psi\left(t - n\frac{T}{2}\right)\ e^{i2\pi\frac{1}{T}mt}. \tag{2.4}$$

The Gaussian pulses in an OQAM-based system are given by

$$g_{k,\ell}(t) \triangleq i^{k+\ell}\ g\left(t - \ell T_\Delta \frac{T}{2}\right)\ e^{i2\pi\frac{F_\Delta}{T}kt}, \tag{2.5}$$

which allows us to write Eqn. (2.2) more compactly as

$$s(t) = \sum_{\ell=-\infty}^{\infty} \sum_{k=0}^{N-1} x_{k,\ell}\ g_{k,\ell}(t). \tag{2.6}$$

The representation of the Gaussian pulse in the basis will be the inner product between $g_{k,\ell}(t)$ and $\psi_{m,n}(t)$ as

$$\begin{aligned} C_{k,\ell,m,n} &\triangleq \langle\ g_{k,l}(t), \psi_{m,n}(t)\ \rangle \\ &= \Re\left\{\int g_{k,\ell}(t)\ \psi^*_{m,n}(t)\ dt\right\}, \end{aligned} \tag{2.7}$$

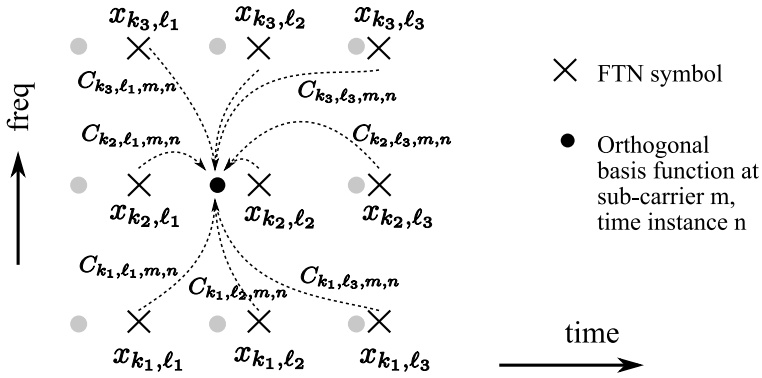

Figure 2.4: Illustration of mapping function on an orthogonal basis function at sub-carrier m and time instance n.

where $C_{k,\ell,m,n}$ represents the projection coefficients of the Gaussian pulse on to the basis (This is also illustrated for one FTN symbol in Figure 2.3). In other words, the coefficients $\{C_{k,\ell,m,n}\}$ represent the interference pattern of an FTN symbol at position (k,ℓ) on a set of orthogonal basis functions in both time and frequency, and Eqn. (2.6) becomes

$$s(t) = \sum_{\ell=-\infty}^{\infty} \sum_{k=0}^{N-1} \sum_{m,n} x_{k,\ell}\ C_{k,\ell,m,n}\ \psi_{m,n}(t). \tag{2.8}$$

The process of representing the FTN symbols in the orthogonal basis is hereafter referred to as *mapping* and a block realizing it is referred to as a *mapper*. A general block diagram of the FTN transmitter using the mapper is shown in Figure 2.2. The mapper produces outputs $x'_{m,n}$ by processing the incoming FTN symbols $x_{k,\ell}$. The number of symbols $x'_{m,n}$ is $T_\Delta F_\Delta$ times the number of symbols $x_{k,\ell}$ for large blocklengths. For a given $T_\Delta F_\Delta$, the projection coefficients $C_{k,\ell,m,n}$ correspond to a unique set of values that can be used to represent all the FTN symbols corresponding to that $T_\Delta F_\Delta$. The FTN mapper evaluates the output at each orthogonal sub-carrier m and time instance n for the incoming FTN symbols together with the help of pre-calculated projection coefficients. The FTN mapper output $(x'_{m,n})$ can be written as

$$\begin{aligned} x'_{m,n} &= x_{k_1,\ell_1} \cdot C_{k_1,\ell_1,m,n} \ + \ x_{k_2,\ell_2} C_{k_2,\ell_2,m,n} + \ x_{k_3,\ell_3} \cdot C_{k_3,\ell_3,m,n} \ + \ \ldots, \\ &= \sum_{p,q} \ x_{k_p,\ell_q} \cdot C_{k_p,\ell_q,m,n}, \end{aligned} \tag{2.9}$$

where $C_{k_p,\ell_q,m,n}$ correspond to the pre-calculated projection coefficients, $x'_{m,n}$ is the value signaled at basis function $\psi_{m,n}(t)$ and x_{k_p,ℓ_q} are the FTN symbols. Eqn. (2.9) is illustrated in Figure 2.4 for one orthogonal sub-carrier and time instance (m, n).

Rectangular windowed sinusoidal basis

For the multicarrier modulation following the mapper, an immediate choice for $\psi(t)$ in Eqn. (2.4) is the rectangular windowed sinusoidal basis (also referred to as rectangular basis for short in this work) in order to use IFFT for multicarrier modulation. The rectangular pulse is optimal in time. However, in frequency it is not very localized and its spectral decay is slow. As a consequence a large set of coefficients are required to represent every Gaussian pulse that carry information implying a significant impact on the FTN transmitter complexity.. If just an IFFT is used for multicarrier modulation, we take $\psi(t) = \text{rect}(t)$ and the transmitted signal will be

$$s(t) = \sum_{n=-\infty}^{\infty} \sum_{m=0}^{N-1} i^{m+n}\ x'_{m,n} \cdot \text{rect}\left(t - n\frac{T}{2}\right)\ e^{i\frac{2\pi}{T}mt}, \tag{2.10}$$

where $x'_{m,n}$ represents the projection of the transmission signal $s(t)$ onto $\psi_{m,n}(t)$.

IOTA basis

It is important to identify a basis that is compact in both time and frequency. This will ensure fewer orthogonal basis functions required to represent an FTN symbol, reducing the computational complexity. A more suitable choice is a pulse such as the Isotropic Orthogonal Transform Algorithm (IOTA) [LFAB95] pulse (c.f Figure 7.1). The IOTA pulse retains the property of time-frequency compactness as it is derived from the Gaussian pulse and is a strong candidate for usage in FTN multicarrier systems. The orthogonal basis generated from the IOTA pulse can be written as

$$\Im_{m,n}(t) = i^{m+n} \cdot e^{i2\pi mft} \cdot \Im\left(t - n\frac{T}{2}\right). \tag{2.11}$$

The IOTA pulse guarantees orthogonality only for real symbols, therefore OQAM systems are considered and the term i^{m+n} in Eqn. (2.11) is the phase offset factor. Eqn. (2.11) refers to multicarrier modulation using OQAM with

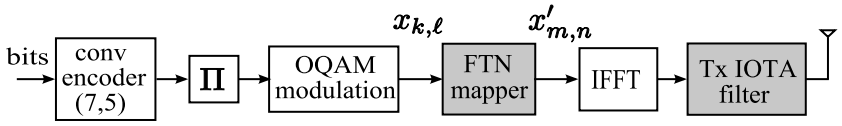

Figure 2.5: Block diagram of the proposed FTN transmitter.

IOTA as the orthogonal basis with time shift $\frac{T}{2}$. By using IOTA as the orthogonal basis, i.e., $\psi(t) = \Im(t)$, the transmitted signal will be

$$s(t) = \sum_{n=-\infty}^{\infty} \sum_{m=0}^{N-1} i^{m+n}\, x'_{m,n} \cdot \Im\left(t - n\frac{T}{2}\right)\, e^{i\frac{2\pi}{T}mt}. \tag{2.12}$$

A block diagram of the FTN transmitter with the mapper and multicarrier modulation using IOTA is shown in Figure 2.5. An outer convolutional code is introduced prior to the FTN mapper to help in better decoding of the FTN modulated symbols at the receiver. The blocks highlighted in grey are those specific to the FTN system. Bypassing these during transmission will result in a multicarrier modulated signal corresponding to a conventional orthogonal (OFDM) system. IOTA in OFDM systems has previously been described in [SSL02, MJ05] where a pulse shaping filter is employed as a post processing block retaining the hardware efficient IFFT as a part of the IOTA multicarrier modulation. The realization of IOTA based multicarrier modulation using IFFT and a pulse shaping filter is derived from Eqn. (2.12) which can be re-written as

$$s(t) = \sum_{n=-\infty}^{\infty} i^n \Im\left(t - n\frac{T}{2}\right) \cdot \sum_{m=0}^{N-1} i^m x'_{m,n} e^{i\frac{2\pi}{T}mt}. \tag{2.13}$$

The term $X''_n(t) \triangleq \sum_{m=0}^{N-1} i^m x'_{m,n} e^{i\frac{2\pi}{T}mt}$ refers to the inverse Fourier transform of the input $i^m x'_{m,n}$, hence

$$\begin{aligned} s(t) &= \sum_{n=-\infty}^{\infty} i^n \Im\left(t - n\frac{T}{2}\right) \cdot X''_n(t) \\ &= \sum_{n=-\infty}^{\infty} \Im\left(t - n\frac{T}{2}\right) \cdot X'_n(t), \end{aligned} \tag{2.14}$$

where $X'_n(t) \triangleq i^n X''_n(t)$. If $X'_n(t)$ is to be represented as a discrete sequence, the inverse Fourier transform should be replaced by an Inverse Discrete Fourier Transform (IDFT). IDFT can be used instead of the inverse Fourier Transform

provided that the number of points of IDFT is equal to or greater than the number of samples of the discrete sequence $x'_{m,n}$ in order to avoid aliasing [PM04]. If the sampling period T_s is related to T the duration of the discrete time sequence and N the number of points of IDFT as $T_s = T/N$ then the transmitted discrete time sequence will be

$$s\left(pT_s\right) = \underbrace{\sum_{n=-\infty}^{\infty} \Im\left(pT_s - n\frac{T}{2}\right)}_{\text{IOTA pulse shaping}} \cdot \underbrace{X'_n\left(pT_s\right)}_{\text{IFFT}}. \tag{2.15}$$

It is also well known that an efficient implementation approach for the IDFT is the IFFT. This implies Eqn. (2.14), which corresponds to an IOTA based multicarrier modulation, can be realized as an IFFT followed by a post filtering operation as indicated in Eqn. (2.15) and Figure 2.5. Choosing IOTA based MCM results in reduced computational complexity. The hardware architecture and implementation of IOTA pulse shaping filter is discussed in greater detail in Chapter 7.

Rectangular vs. IOTA basis

In the following we will compare the IOTA and rectangular basis with respect to the requirements on the number of basis functions required across time (N_t) and frequency (N_f) to represent an FTN pulse. The aim is to obtain a reasonable number for N_t and N_f such that the representation is realized by a small set of basis functions and at the same time the reconstruction would be as close to the original FTN pulse as possible. The FTN pulse $g_{k,\ell}(t)$ can be represented using $N_t \times N_f$ projection coefficients $C_{k,\ell,m,n}$ as

$$g_{k,\ell}(t) \approx \sum_m \sum_n C_{k,\ell,m,n}\ \psi_{m,n}(t). \tag{2.16}$$

As N_t and N_f become large, the FTN pulse representation becomes more accurate but at the same time introduce higher computational complexity. The investigated combinations of $N_t \times N_f$ are 2×2, 3×3, 3×2, 3×4, 5×5 and 9×9 for both IOTA and rectangular basis. Higher values of $N_t \times N_f$ are not attractive for low complexity implementations. Furthermore, N_t and N_f are not dependent on $T_\Delta F_\Delta$ but on the choice of the basis pulse. This is because, irrespective of the spacing between the adjacent pulses, each pulse in itself has to be represented with the best possible accuracy. Figure 2.6 shows a comparison between the FTN pulses reconstructed from both IOTA and rectangular windowed sinusoidal basis for two cases, $N_t \times N_f = 3 \times 3$ and 9×9. The upper plot in Figure 2.6 shows the original and the reconstructed FTN

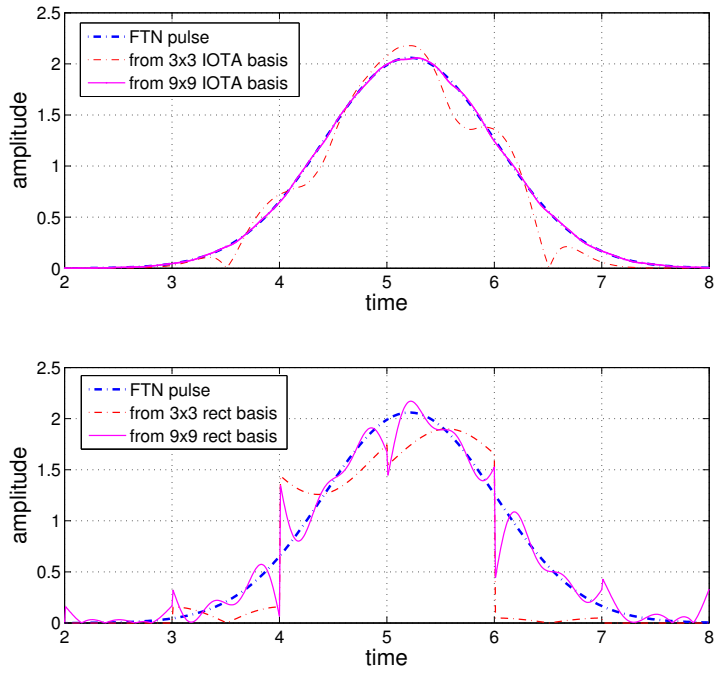

Figure 2.6: Comparison of FTN pulse (of time period 3T) reconstructed from IOTA and rectangular windowed sinusoidal basis.

pulse from an IOTA basis using 3×3 and 9×9 basis functions, while the lower plot shows the reconstruction using the same combination from the rectangular basis. The reconstruction using 3×3 IOTA basis functions is fairly good while with 9×9, the reconstruction is indistinguishable as it overlaps on the original pulse. However, the reconstruction from 3×3 rectangular windowed sinusoidal basis functions is not satisfactory, while the 9×9 is better but is still worse than the 3×3 IOTA. This argument is further supported from the frequency response plots of the reconstructed pulses. Figure 2.7 shows the spectrum of the pulses reconstructed from IOTA and rectangular basis on a particular sub-carrier. The ICI introduced by representing the Gaussian pulse in the IOTA basis is far less compared to the rectangular basis. Further, the main lobe of the IOTA reconstruction is same as the original FTN pulse, while the one from rectangular basis has poor reconstruction as well as significant ICI. The other

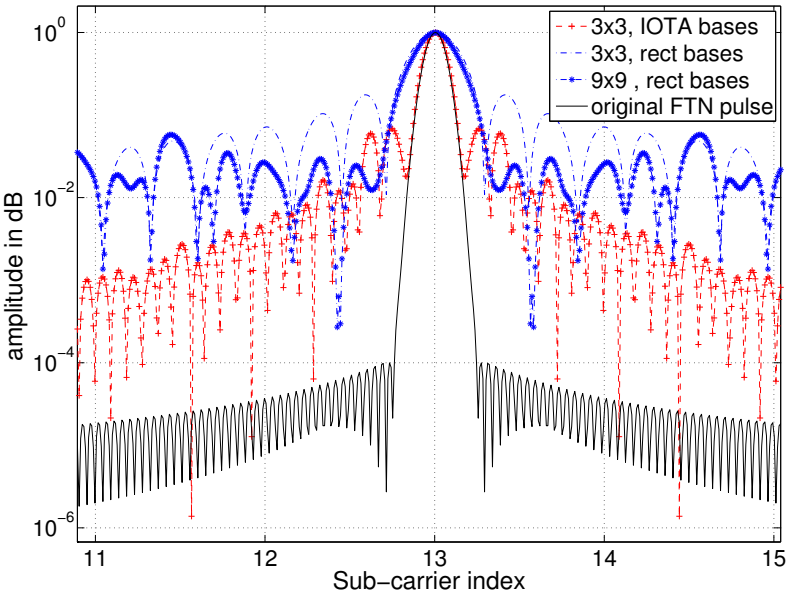

Figure 2.7: Spectrum of reconstructed pulse on a certain sub-carrier.

evaluated combinations were discarded for the following reasons:

i) 2×2 projections were not satisfactory in either IOTA or rectangular windowed sinusoidal basis. The number being even did not distribute itself uniformly over the FTN symbol either in time or frequency. Hence one side of the FTN symbol energy was lost in most cases resulting in poor representation of the pulse.

ii) 3×2 projections produced fair reconstructions in a few cases, but failed in certain configurations.

iii) 3×4 projections gave satisfactory results but not considered due to the extra processing requirements compared to the 3×3 configuration.

Furthermore, IOTA filtering has been used in orthogonal multicarrier systems to avoid the cyclic prefix [LFAB95, JL03] and is now a part of the 3GPP standard [3GP04]. The effective overhead due to FTN signaling is mainly due to the mapper. By using the IOTA, the number of projections for each pulse $g_{k,\ell}(t)$ can be as low as 3×3 [DRAO09]. Due to the fractional spacing, the FTN

pulse may be represented by the $N_t \times N_f$ basis functions at varying accuracies. With 3 sub-carriers and 3 time instances of the basis the best represented pulse retains up to 99.5% of the energy of an FTN pulse while the least accurate representation preserves 87% of the energy.

2.2 Alternate transmission methods

This section briefly mentions two other ways by which FTN signaling can be realized, and their disadvantages when considered for hardware realization.

Method1

Fractional fast Fourier transforms [BS91] can be used to realize Eqn. (2.2) as is by having $T_\Delta = 1$ and $F_\Delta < 1$. In [BS91], the fractional Fourier transform of a sequence $\mathbf{x}$ is defined as

$$X_k(\mathbf{x}, \alpha) = \sum_{n_{\text{frac}}=0}^{N_{\text{frac}}-1} x_{n_{\text{frac}}} e^{i2\pi n_{\text{frac}} k\alpha}, \tag{2.17}$$

where the parameter α defines the fractional spacing, N_{frac} the length of the transform and all other variables having their conventional meaning as in a Fourier transform [PM04]. Apparently, it seems that a single module of fractional FFT will be sufficient. However, if one would like to have different $T_\Delta F_\Delta$ in order to achieve varying bandwidth efficiencies, then the fractional FFT should be capable of carrying out the transform for different fractional spacings. Furthermore, if in a system the sub-carrier length varies the fractional FFT module should also be capable of such an adaptation. Thus, the fractional FFT should be reconfigurable in both number of points of the transform as well as the fractional spacing between the frequencies of the transform. Such a fractional FFT might not have the regularity that is found in conventional FFTs, using radix-2, radix-4 [PM04] etc., as the building blocks for the transform. Lastly, fractional FFTs brings up the requirement of a specialized multicarrier modulation scheme resulting in a system that requires drastic changes in design and implementation. Thus, it deviates significantly from conventional implementations limiting co-existence between FTN and orthogonal systems.

Method2

In the second alternative approach, multiple IFFTs can be used to obtain the FTN modulated signal. For example, an FTN modulated signal where $T_\Delta F_\Delta = 0.5$ with $T_\Delta = \frac{1}{2}$ and $F_\Delta = 1$ can be evaluated as

$$s(t) = \mathcal{F}^{-1}\{X_1(f)\} + \mathcal{F}^{-1}\{X_2(f)\ e^{-i2\pi f \frac{T}{2}}\} \tag{2.18}$$

$$s(t) = x_1(t) + x_2\left(t - \frac{T}{2}\right), \tag{2.19}$$

and an FTN system with $T_\Delta F_\Delta = \frac{1}{3}$ would require 3 IFFTs, evaluated as

$$s(t) = \mathcal{F}^{-1}\{X_1(f)\} + \mathcal{F}^{-1}\{X_2(f)\ e^{-i2\pi f \frac{T}{3}}\} + \mathcal{F}^{-1}\{X_3(f)\ e^{-i2\pi f \frac{2T}{3}}\}$$

$$s(t) = x_1(t) + x_2\left(t - \frac{T}{3}\right) + x_3\left(t - \frac{2T}{3}\right). \tag{2.20}$$

The above discussed alternative was proposed in [DRO11] and realizes FTN by signaling faster in time. A similar approach proposed in [WPID11] realizes FTN by squeezing the sub-carriers closer together with $T_\Delta = 1$ and $F_\Delta = \frac{1}{2}$. In that case, the transmitted signal would be

$$s(t) = \mathcal{F}^{-1}\{X_1(f)\} + \mathcal{F}^{-1}\left\{X_2\left(f - \frac{1}{2T}\right)\right\} \tag{2.21}$$

$$s(t) = x_1(t)\ +\ x_2(t)\, e^{i\pi t \frac{1}{T}}. \tag{2.22}$$

Eqns. (2.18) and (2.19) form the time shift property of the Fourier transform while Eqns. (2.21) and (2.22) correspond to the frequency shift/complex modulation property [PM04].

[WPID11] achieves FTN by having information carrying symbols closer in frequency as in Eqn. (2.21). This is achieved by the use of multiple IFFTs and some post-processing in the form of complex phase rotators. Further, the complex phase rotators that need to be multiplied in Eqn. (2.22) vary depending on the choice of frequency spacing.

The alternative proposed in [DRO11] achieves FTN by signaling faster in time and also requires multiple IFFTs. This approach needs pre-processing before carrying out the IFFTs as shown in Eqn. (2.18).

The multiple IFFTs approach suffer from some drawbacks when it comes to overhead complexity in realizing FTN signaling. Changing the FTN parameters, such as time-frequency spacing, requires substantial changes in hardware such as the requirement of several FFTs that take up significant hardware resources. Further, varying number of sub-carriers in the system brings in

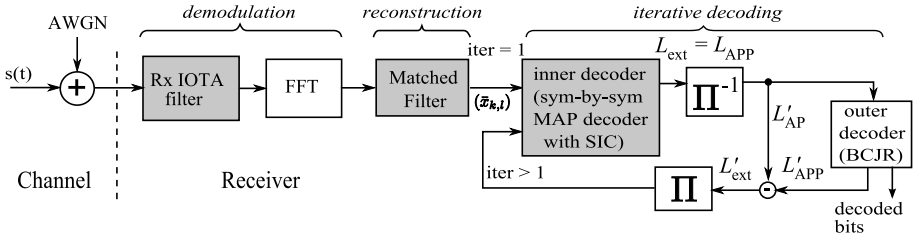

Figure 2.8: Block diagram of the FTN receiver.

additional complexity. Though these factors can be accounted by realizing the worst case scenario for the system, it poses a risk of very large hardware overhead.

2.3 Decoding FTN modulated symbols

The decoding scheme proposed in this article is comprised of an iterative detector with Matched Filter (MF) and Successive Interference Cancellation (SIC). The decoding algorithm is primarily conceived in [RA09b] but has been modified for hardware efficient implementation in this work by re-use of processing blocks in different scenarios. Figure 2.8 shows the block diagram of the receiver and custom blocks specific to FTN signaling are highlighted in grey. The subsections described henceforth are ordered in the way the received signals are processed. The modulated signal $s(t)$ is assumed to be transmitted through an AWGN channel as shown in Figure 2.8. The multicarrier demodulation is assumed to be carried out in the same way as the two step approach of multicarrier modulation in the transmitter discussed in Section 2.1.1 under '**IOTA basis**'.

2.3.1 Matched Filtering for FTN symbol reconstruction

The multicarrier demodulated signal represents the transmitted projections of FTN symbols affected by an AWGN channel $\eta_{m,n}$. In order to decode the received FTN modulated block into binary information, it has to be reconstructed back from the projections. The approximation of the original FTN symbol is obtained from the received symbols and the pre-computed unique projection coefficients for that FTN symbol (interference pattern $C_{k,\ell,m,n}$) using an MF.

The MF in Figure 2.8 reconstructs the FTN symbols as

$$\bar{x}_{k,\ell} = \sum_{m,n} C_{k,\ell,m,n} \quad \left(x'_{m,n} + \eta_{m,n}\right), \tag{2.23}$$

where $x'_{m,n}$ are the accumulated projections of FTN symbols transmitted over an AWGN channel. $C_{k,\ell,m,n}$ correspond to the same pre-computed projection coefficients that is used in the transmitter. The so reconstructed FTN symbols, $\bar{x}_{k,\ell}$, are not free from ISI/ICI as they were originally non-orthogonal when transmitted. As a result, the reconstructions comprise information of the symbol of interest, interference (both ISI and ICI) together with a noise component. The process of matched filtering results in the noise being colored and is denoted as $\eta'_{k,\ell}$. Eqn. (2.23) becomes

$$\bar{x}_{k,\ell} = \sum_{m,n} C_{k,\ell,m,n} x'_{m,n} + \eta'_{k,\ell}. \tag{2.24}$$

Substituting Eqn. (2.9) for $x'_{m,n}$ in Eqn. (2.24) we get

$$\begin{aligned} \bar{x}_{k_1,\ell_1} = \sum_{m,n} C_{k_1,\ell_1,m,n} \cdot \Bigg(& \underbrace{x_{k_1,\ell_1}\ C_{k_1,\ell_1,m,n}}_{\text{signal component at } k_1,\ell_1} \\ & \underbrace{+ \sum_{(k_p,\ell_q)\neq(k_1,\ell_1)} x_{k_p,\ell_q}\ C_{k_p,\ell_q,m,n} \Bigg) + \eta'_{k_1,\ell_1}}_{\text{interference+noise at index } k_1,\ell_1} . \end{aligned} \tag{2.25}$$

From Eqn. (2.25), it can be seen that the reconstructed FTN symbol $\bar{x}_{k_1,\ell_1}$ at sub-carrier k_1 and time instance ℓ_1, has signal component x_{k_1,ℓ_1} as well as interferences from neighboring symbols $x_{k_2,\ell_2},\ x_{k_3,\ell_3},\ \ldots$, and colored noise η'_{k_1,ℓ_1}. The signal component from the reconstructed symbols can be obtained through iterative decoding. The proposed decoder consists of symbol-by-symbol sub-optimal maximum aposteriori (MAP) decoding with successive interference cancellation as the inner decoder and a standard BCJR for the convolutional encoder[1] as shown in Figure 2.8.

[1]The codes used in this paper are low memory codes so that the BCJR is of practical complexity.

2.3.2 Inner Decoder

The inner decoder consists of soft output calculation, SIC and LLR calculation blocks as shown in Figure 2.9. The blocks within the inner decoder are described in the following subsections in the order of their processing.

Soft Output Calculation

The soft output calculation block evaluates the soft symbols from the log-likelihood ratios (LLRs) received from the outer decoder as

$$\begin{aligned}\tilde{x}_{k,\ell} &= \{+1 \cdot \mathrm{P}(x_{k,\ell} = +1)\} + \{-1 \cdot \mathrm{P}(x_{k,\ell} = -1)\},\\ &= \mathrm{P}(x_{k,\ell} = +1) - \mathrm{P}(x_{k,\ell} = -1),\\ &= (1 - \mathrm{P}(x_{k,\ell} = -1)) - \mathrm{P}(x_{k,\ell} = -1),\\ &= 1 - \frac{2}{1 + e^{\mathrm{LLR}(x_{k,\ell})}}, \end{aligned} \tag{2.26}$$

where $x_{k,\ell}$ are OQAM modulated symbols. The Likelihood ratio (LR) and LLR are assumed to have the following definitions:

$$\text{Likelihood Ratio}(x_{k,\ell}) = \frac{\mathrm{P}(x_{k,\ell} = +1|\mathbf{x})}{\mathrm{P}(x_{k,\ell} = -1|\mathbf{x})}, \tag{2.27}$$

and

$$\mathrm{LLR}(x_{k,\ell}) = \ln\left(\frac{\mathrm{P}(x_{k,\ell} = +1|\mathbf{x})}{\mathrm{P}(x_{k,\ell} = -1|\mathbf{x})}\right), \tag{2.28}$$

where $\mathbf{x}$ represents the received sequence. From Eqn. (2.28) it can be deduced that

$$\mathrm{P}(x_{k,\ell} = -1|\mathbf{x}) = \frac{1}{1 + e^{\mathrm{LLR}(x_{k,\ell})}}. \tag{2.29}$$

Successive Interference Cancellation

The concept of SIC is well known and widely used in the the field of communications and is applicable to a broad class of problems in the field. In a multiuser system, it can used to extract individual user data [Ver98, RSAA98, WP99]. In the present context of FTN, SIC is used to repeatedly cancel out the induced ISI and ICI arising due to FTN signaling.

The processing blocks in SIC for FTN signaling are shown inside the dashed box in Figure 2.9. The SIC blocks are re-used from those designed specifically

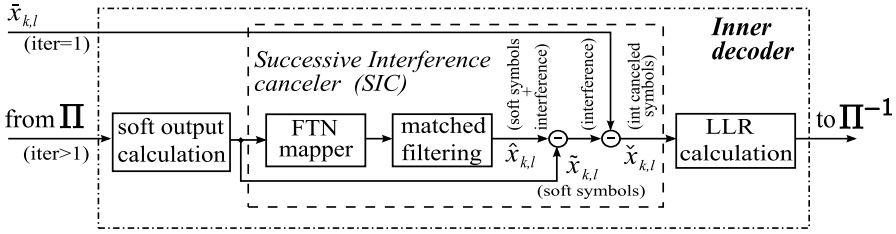

Figure 2.9: Inner decoder and its component processing blocks.

for FTN signaling, i.e., FTN mapper and MF. The symbols $\hat{x}_{k,\ell}$ at the output of the mapper−MF cascade (Figure 2.9) represents the amount of information the soft symbol holds and the interference it experiences from the neighboring symbols (Eqn. (2.25)) due to FTN signaling. Eqn. (2.25) can be expressed more generally as

$$\hat{x}_{k,\ell} = \overbrace{\tilde{x}_{k,\ell}}^{\text{soft symbol}} + \overbrace{\sum_{(p,q)\neq(k,\ell)} \tilde{x}_{p,q} + \eta'_{k,\ell}}^{\text{interference + noise}}, \tag{2.30}$$

where $\tilde{x}_{k,\ell}$ is the signal component and $\tilde{x}_{p,q}$ is the interference component. The soft outputs ($\tilde{x}_{k,\ell}$) when subtracted from its corresponding output at the mapper-matched filter cascade (Eqn. (2.30)) leaves behind the total interference plus noise that the soft symbol experiences and is given as

$$\hat{x}_{k,\ell} - \tilde{x}_{k,\ell} = \sum_{(p,q)\neq(k,\ell)} \tilde{x}_{p,q} + \eta'_{k,\ell}. \tag{2.31}$$

Once an estimate of the interference plus noise is calculated, it can be readily canceled out from the received symbols to leave behind a cleaner signal component and is given as

$$\overbrace{\check{x}_{k,\ell}}^{\text{int. canc. symbol}} = \overbrace{\bar{x}_{k,\ell}}^{\text{reconst. symbol}} - \overbrace{\{\hat{x}_{k,\ell} - \tilde{x}_{k,\ell}\}}^{\text{estimate of int.+noise}}, \tag{2.32}$$

where $\check{x}_{k,\ell}$ represents the interference canceled symbols and $\{\hat{x}_{k,\ell} - \tilde{x}_{k,\ell}\}$ the estimate of the interference plus noise. $\bar{x}_{k,\ell}$ represents the received symbols at the output of the first MF after the FFT (in Figure 2.8). The interference canceled symbols ($\check{x}_{k,\ell}$) are now used as the new set of received symbols to calculate the LLRs in the inner decoder. During each new iteration of decoding in the inner decoder, the interference estimates improve and SIC cleans the reconstructed FTN symbols of noise and interference resulting in better decoding performance.

LLR calculation

The calculation of the LLRs in the inner decoder is derived with the assumption of additive white Gaussian noise. It turns out that the equation to calculate LLRs is very simple for the symbol-by-symbol MAP decoding (and can also be found in [LC04]). The LLRs are calculated as

$$\mathrm{L_{ext}}\{\check{x}_{k,\ell}\} = \ln\left\{\frac{\mathrm{P}(\check{x}_{k,\ell}=+1|\mathbf{x})}{\mathrm{P}(\check{x}_{k,\ell}=-1|\mathbf{x})}\right\}, \tag{2.33}$$

which simplifies to

$$\mathrm{L_{ext}}\{\check{x}_{k,\ell}\} = \frac{2\ \check{x}_{k,\ell}}{\sigma^2_{N+I}}, \tag{2.34}$$

where $\check{x}_{k,\ell}$ refers to the interference canceled symbols of block size $K(=M\times N)$. The variance estimate of the noise plus interference (σ^2_{N+I}) from FTN signaling is calculated as

$$\begin{aligned}\sigma^2_{N+I} &= \mathrm{var}(\check{x}_{k,\ell}) - \mathrm{var}(x_{k,\ell}) \\ &\approx \frac{1}{K}\sum_{k,\ell}\check{x}^2_{k,\ell} - \frac{1}{K}\sum_{0}^{K-1} 1 \\ &= \frac{1}{K}\sum_{k,\ell}\check{x}^2_{k,\ell} - 1.\end{aligned} \tag{2.35}$$

In the first iteration, the inner decoder calculates the LLRs using the symbols reconstructed from matched filtering as there is no estimate of the interference. In subsequent iterations it is done on the interference canceled symbols. The inner decoder block directly calculates the extrinsic LLRs from the input LLRs. As a result, apriori LLRs need not be subtracted at the output of the inner decoder (see Figure 2.8).

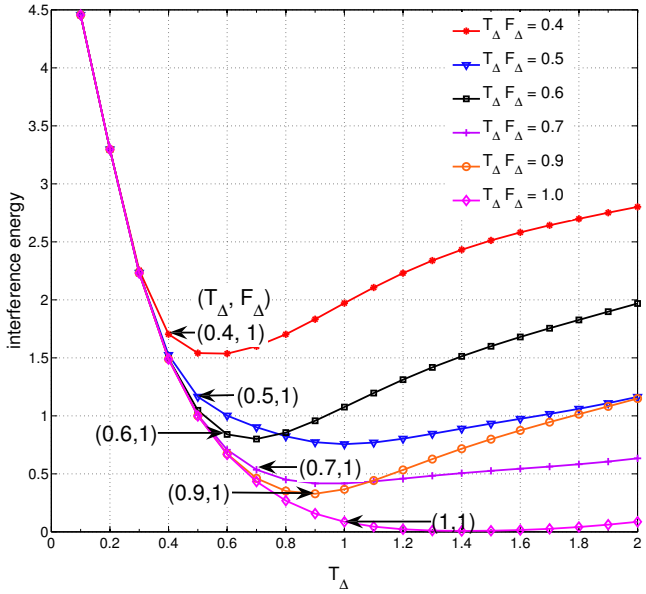

Figure 2.10: Plot showing the interference energy curves for different time-frequency spacings $T_\Delta F_\Delta$.

2.4 Choice of Time-Frequency spacing in FTN signaling

Prior to presenting the performance of the proposed receiver, we establish the reason for choosing a particular time and frequency spacing (T_Δ, F_Δ). Strictly speaking, though individual values of T_Δ or F_Δ are important, it is the product $T_\Delta F_\Delta$ that defines the effective improvement in bandwidth usage. As a result, for any given product that attains a certain amount of bandwidth efficiency, T_Δ and F_Δ can theoretically take infinite number of values. However, it is important to choose the spacing such that the interference by the FTN symbols to its neighbors is minimal compared to the rest of the possible spacing alternatives. Figure 2.10 shows the plots of interference energy of FTN symbols to its neighbors as a function of the time spacing T_Δ. The interference energy (E_{int}), on an FTN pulse at a certain index (k_1, ℓ_1) from a set of surrounding

FTN pulses $g_{k,\ell}(t)$, as a function of T_Δ for a given product $T_\Delta F_\Delta$ is given by

$$E_{\text{int}}(T_\Delta)|_{T_\Delta F_\Delta=\text{P}} = \underbrace{\sum_{k,\ell} |\langle g_{k,\ell}(t), g_{k_1,\ell_1}(t)\rangle|^2}_{\text{total energy at } (k_1,\ell_1)} - \underbrace{|\langle g_{k_1,\ell_1}(t), g_{k_1,\ell_1}(t)\rangle|^2}_{\text{energy of symbol at } (k_1,\ell_1)}, \tag{2.36}$$

where $\text{P} = \{0.4, 0.5, 0.6, 0.7, 0.9, 1.0\}$ and $g_{k,\ell}(t)$ refers to a Gaussian pulse $g(t)$ at sub-carrier k and time instance ℓ[2]. The first term on the right hand side of Eqn. (2.36) gives the total interference of all FTN pulses, $g_{k,\ell}(t)$, on the FTN pulse at (k_1, ℓ_1). From this the second term, which represents the energy on the FTN pulse at (k_1, ℓ_1), is subtracted to obtain the total interference. Each curve represents a fixed time-frequency spacing ($T_\Delta F_\Delta \leq 1$) with interference energy along y-axis, as we vary T_Δ along the x-axis. From the figure it is evident that when spacing T_Δ is very small implying that the symbols are very close to each other the total interference on any symbol from its neighbors is very high. Similar is the case when T_Δ is large (because F_Δ turns out to be very small). These extreme values can be ignored outright as a large amount of induced interference will have an impact on the decoding performance. The optimal operating points would be those where the interference energy is at its least. At these optimal points, the induced ISI and ICI due to FTN signaling will be minimal for the given product $T_\Delta F_\Delta$ and helps in better decoding at the receiver. We strongly believe that this optimizes the system. However, the system is operated slightly away from the optimal operating points such that $F_\Delta = 1$ always[3], as indicated in Figure 2.10. With this, by simply changing the parameter T_Δ, different $T_\Delta F_\Delta$ can be accomplished.

For the $T_\Delta F_\Delta = 0.5$ curve, an operating point at $T_\Delta = 1$ is a more appropriate choice in the context of minimal interference than the chosen point at $T_\Delta = 0.5$. However, this has not been considered keeping in mind the hardware implementation. A homogeneous choice of T_Δ and F_Δ results in simplified computational units as well as the control unit that manages the data flow. It will also be shown later that the chosen operating points result in a more efficient hardware implementation inspite of the higher interference.

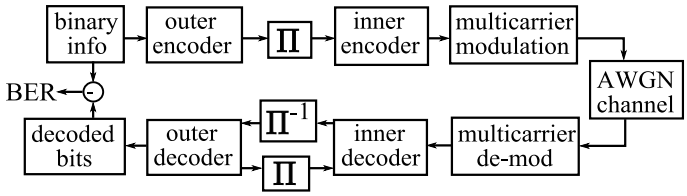

Figure 2.11: System setup of FTN transceiver to evaluate the decoder performance.

2.5 System setup

The performance of the FTN decoder is evaluated in MATLAB by using the system setup shown in Figure 2.11. The sequence of blocks **outer encoder - interleaver - inner encoder - multicarrier modulation** correspond to the FTN transmitter shown in Figure 2.5. The so generated symbols are transmitted through an AWGN channel. The receiver chain is a generalization of Figure 2.8. We measure the performance in terms of the SNR, average energy per bit over N_0 (E_b/N_0). Random interleaving is assumed during the simulations. The information block size is chosen by specifying the number of sub-carriers N and time instances M. For simulations, a system with 1000 sub-carriers for 20 time instances was chosen as the block size. A total of about 1M information bits are decoded for each SNR and per FTN configuration (T_Δ). The number of decoding iterations in the setup is also parameterizable but is fixed at 8.

2.6 Receiver performance

Figure 2.12 presents the decoding performance of the FTN receiver for different time spacings (T_Δ). It can be seen that at higher SNRs, the receiver performance is quite good and approaches the theoretical limit corresponding to the BER curve for the $(7, 5)$ convolutional code ($T_\Delta = 1.0$) in an interference free Gaussian channel [Rus07, Lee01]. At lower SNRs the deviation is significant as the induced interference is hard to clean when the symbols are stacked close together. For SNR = 5dB, the FTN system using $T_\Delta = 0.4$ still has bad performance compared with lower SNR levels, while for $T_\Delta \geq 0.5$ it improves significantly. It is evident from Figure 2.12 that one could employ the FTN sys-

[2]It is to be noted here that the indices k, ℓ refers to the sub-carrier and time instances on FTN/non-orthogonal grid

[3]Since F_Δ is fixed at 1, the FTN signaling parameter $T_\Delta F_\Delta$ is sometimes simply referred to by T_Δ

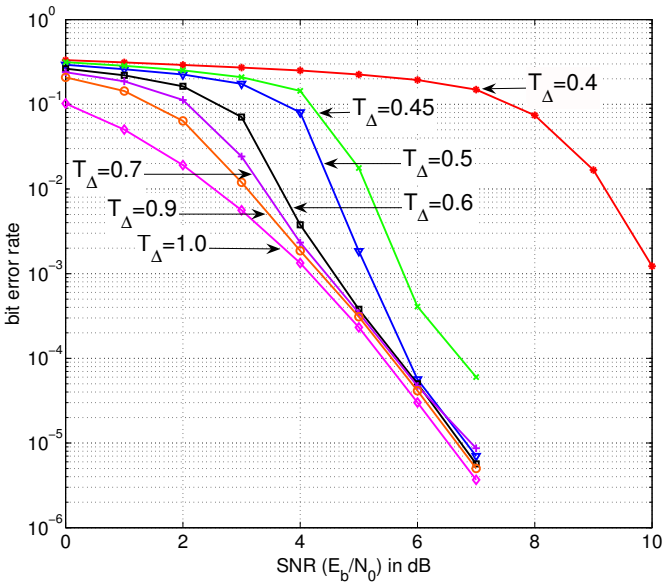

Figure 2.12: Receiver performance for varying time spacings T_Δ with 3x3 projection coefficients.

tem to achieve higher bandwidth efficiency when channel conditions are good. The FTN system provides 2x improvement in bandwidth efficiency when compared to an OFDM system using the same modulation at the cost of extra complexity in receiver processing. It was found that 8 iterations are sufficient to generate close to optimal results in terms of the decoding performance.

Our work presented in [DRAO09] has shown that the number of projection points is crucial for the overhead in transmitter complexity. The case is similar for the receiver since the same mapper is used in the SIC. The MF, the other component of SIC, also has the same order of computational complexity per FTN symbol as that of the FTN mapper. Hence the number of projection coefficients play a major role in the receiver complexity as well. Ideally, all projection coefficients have to be considered to completely reconstruct the Gaussian pulse carrying FTN symbols, hereby simply referred to as an FTN pulse.

In Section 2.1.1 it was shown that in order to represent the FTN pulse fairly accurately, 3×3 projection coefficients are required when using the IOTA pulse. However, this does not say much about the degradation in receiver

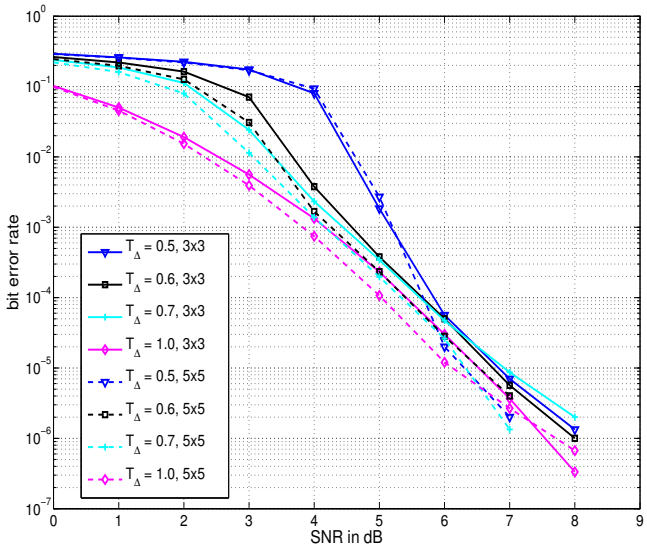

Figure 2.13: Bit Error Rate comparison between 3x3 and 5x5 projection coefficients with $T_\Delta = \{0.5, 0.6, 0.7 \text{ and } 1.0\}$.

performance. In order to evaluate the impact of using different number of projection coefficients, the receiver is evaluated for configurations 3×3 and 5×5.

The receiver is simulated using the system setup shown in Figure 2.11. The results are presented in Figure 2.13 showing the performance comparison when the FTN system is operated at different spacings (T_Δ) at varying SNRs and projection points being 3×3 and 5×5. The solid line is the performance for 3×3 projection points, while the dashed lines show the performance for 5×5 with all other parameters remaining the same. Using 5×5 projection coefficients requires 25 operations per FTN symbol, while the 3×3 requires only 9. The performance curves from Figure 2.13 show that the improvement from 5×5 coefficients is only marginal when compared to 3×3. This shows that the complexity is kept moderate by using 3×3 projection coefficients with only small degradation in performance.

Going beyond 5×5 is computationally intensive and prohibitive when considering hardware implementations. Though 9×9 projection coefficients was evaluated for the FTN transmitter, it was for this reason that it was not consid-

ered for evaluation in the receiver. It can also be noted that 9×9 coefficients was the minimum requirement when using rectangular basis, indicating the inappropriateness of its choice for FTN.

2.6.1 Finite wordlength considerations.

A step closer towards implementation of the FTN receiver in hardware is to determine the finite wordlength requirements in the processing blocks. This is done by evaluating the receiver model under fixed point considerations. The iterative decoder passes the LLRs back and forth between the two component decoders. These LLRs are numbers represented in the log domain. Hence when performing calculations on the LLRs, it may be sufficient to represent them using fewer bits, or in other words, cover a small dynamic range with respect to the LLRs. Due to the log scale, a small dynamic range for the LLRs corresponds to a very wide dynamic range in the actual scale. Table 2.1 shows evaluated fixed point representation in this work and two other works from literature [MB00, MWW00].

Table 2.1: Comparison showing wordlengths evaluated in this work and two other works from literature.

	Integer wordlength	Fractional wordlength
[MB00]	3	2-3
[MWW00]	5	3-4
This work	4	2,4,6

The chosen wordlengths are primarily used for the representation of LLRs. The same wordlength is also used within the inner decoder to represent the soft symbols and the values during SIC. Higher precision helps in better representation of these soft symbols when SIC is carried out. The range of the LLRs is set to ± 5 as the decoder rarely reverts from high LLR values. To cover this dynamic range 4 bits is sufficient and hence the integer part is fixed at 4 bits. The fractional wordlength was determined by evaluating the receiver performance using 2, 4 and 6 precision bits. At all interfaces between the blocks depicted in Figures 2.5, 2.8 and 2.9 the inputs are quantized with the above mentioned wordlengths. Internal to the blocks, full precision is maintained and the final result is quantized to same the wordlength as the input and are passed on to the subsequent blocks.

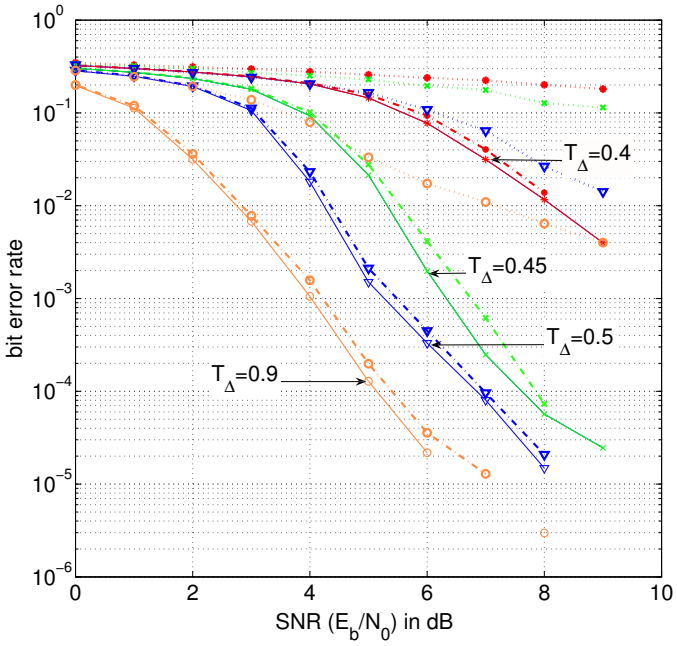

Figure 2.14: Performance of fixed point receiver model (dash-dot lines: 8 bit, dotted lines: 6 bit wordlength) in comparison with floating point benchmark (solid line).

Figure 2.14 shows the performance from fixed point simulations with wordlengths of 6 bits and 8 bits in comparison with the floating point performance. The solid lines correspond to the BER curves of the model using floating point precision, while the dash-dotted and dotted curves represent the BER performance with fixed point representation using 8 bits and 6 bits respectively. It can be seen that the performance of the fixed point model using 8 bits closely follows the full precision counter part, while that using 6 bits is significantly farther away from the ideal curve. The performance with 8 bits deviated from the floating point performance only by a few fractions of a dB. Increasing the wordlength further to 10 bits will only increase the hardware complexity with negligible improvement in performance.

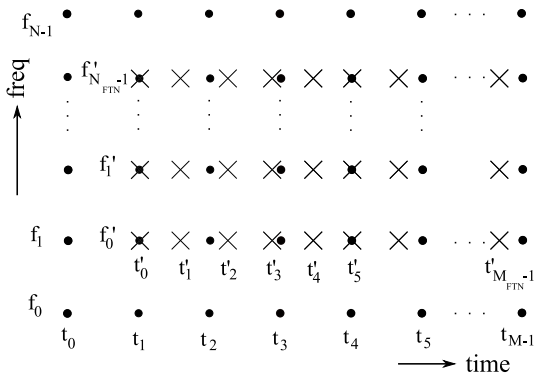

Figure 2.15: General grid of FTN and OFDM symbols.

2.6.2 Fixing the block size for interleaver/de-interleaver design.

In this work, the size of the information block is chosen to be 2016, a specification from the 3GPP standard [3GP07]. This choice of block size is to show compatibility of FTN system with standardized systems as well as to use the well defined specifications for hardware implementation.

The performance of a system with a block size of about 1000 has a margin of $1-2$ dB worse than that employing a block size of 10^4 or 10^5 [BDMP98]. Comparing the performance degradation to the latency and memory requirements for handling large block sizes of 10^4, a block size of ≈ 2000 is reasonable from an implementation point of view.

2.7 Gains from the FTN system

In a multicarrier system with N sub-carriers and information transmitted in blocks of $N \times M$, M being the number of time instances, the equivalent number of FTN symbols will be greater than $N \times M$ with FTN signaling since $T_\Delta F_\Delta < 1$. If the FTN symbols are projected onto N_t time instances and N_f sub-carriers, the effective number of symbols transmitted is given by

$$N_{\mathrm{FTN}} \times M_{\mathrm{FTN}} = \left(\left\lceil \frac{N-(N_f-1)}{F_\Delta} \right\rceil - 1 \right) \times \left(\left\lceil \frac{M-(N_t-1)}{T_\Delta} \right\rceil - 1 \right), \quad (2.37)$$

where N_{FTN} and M_{FTN} are the actual number of sub-carriers and time in-

stances of the FTN system. In Eqn. (2.37), when $F_\Delta = 1$

$$N_{\text{FTN}} = N - (N_f - 1), \tag{2.38}$$

and when $T_\Delta = 1$

$$M_{\text{FTN}} = M - (N_t - 1). \tag{2.39}$$

In order to allow for FTN symbols close to the sub-carrier and temporal boundaries of a block of information to be represented with the same accuracy as others, some sub-carriers are to be reserved along these boundaries. Figure 2.15 illustrates the orthogonal basis functions (•s) that span a larger number of sub-carriers and time instances than the FTN symbols (×s). The number of sub-carriers and time instances that needs to be reserved are

$$\text{No. of reserved sub}-\text{carriers} = (N_f - 1) \times M, \tag{2.40}$$

and

$$\text{No. of reserved time instances} = (N_t - 1) \times (N - (N_f - 1)). \tag{2.41}$$

Since the FTN symbols are transmitted by projecting them onto the orthogonal basis functions, the number of reserved resources depends on the number of projection points. This has been factored in Eqn. (2.37) that evaluates the total number of FTN symbols that can be transmitted using a given amount of time-frequency resource and FTN signaling parameter $T_\Delta F_\Delta$. With 3×3 projections, $2M$ sub-carriers and $2(N-2)$ time instances are to be reserved (using Eqns. (2.40), (2.41) and $N_t = N_f = 3$). When the FTN symbols are represented using a larger set of projection coefficients, say 9×9, $8M$ sub-carriers and $8(N-8)$ time instances are to be reserved ($N_t = N_f = 9$). In practice, the information block size transmitted over a wireless channel varies from a few hundred to a few tens of thousand bits. For example, in a block size of 10 000 assuming equal distribution of resources across frequency and time there will be 100 sub-carriers and time instances. When $N_t = N_f = 3$ and from Eqns. (2.40),(2.41), a total of 200 sub-carriers and 196 time instances need to be reserved and hence will be unused. Although $N_t = N_f = 9$ results in better FTN pulse representation, it requires 800 sub-carriers and 736 time instances as reserved resources. With smaller block sizes the ratio between used to the unused resources gets worse when N_t and N_f are large. The reservation of these resources should not offset the gains that are achieved through FTN signaling. This is yet another reason for choosing small values of N_t and N_f. The increase in the number of symbols that can be transmitted using the FTN technique becomes

$$\frac{N_{FTN} \times M_{FTN}}{N \times M} \rightarrow \frac{1}{T_\Delta F_\Delta}. \tag{2.42}$$

This means that with small values of N_t, N_f and sufficiently large M and N, one tends towards reaching the theoretical limit of what could be achieved for the given configuration of the FTN system.

Table 2.2 gives a comparison of the actual number of coded symbols transmitted in the presented FTN system as compared to a conventional multicarrier (OFDM) system with the following system parameters: $N = 128$, $M = 16$ and $F_\Delta = 1$. It can be seen that the number of information symbols that can be transmitted using the FTN technique can be nearly doubled compared to its orthogonal counterpart. The theoretical limit can be approached by using larger block sizes. Further, the compression factors T_Δ and F_Δ can be chosen depending on the channel conditions varying all the way from $T_\Delta F_\Delta = 1$ to $T_\Delta F_\Delta = 0.4$ or even further until one reaches the ultimate limit [RA09b] of packing the symbols with a tradeoff in receiver complexity.

Table 2.2: FTN symbols transmitted in a system with block size 128×16.

T_Δ	$N_{\text{FTN}} \times M_{\text{FTN}}$	total symbols	actual ratio (using Eqn. (2.37))	theoretical ratio ($\frac{1}{T_\Delta F_\Delta}$)
0.4	126×34	4284	2.09x	2.50x
0.45	126×31	3906	1.90x	2.22x
0.5	126×27	3402	1.66x	2.00x
0.6	126×23	2898	1.41x	1.60x
0.7	126×19	2394	1.16x	1.42x
0.9	126×15	1890	0.92x	1.11x

2.8 Summary

An algorithmic perspective of FTN signaling, taking into account aspects of hardware complexity, in the transmitter and the receiver is detailed in this chapter. At the transmitter, the choice of orthogonal basis and its influence on FTN complexity is presented. The mapper based approach in the transmitter will introduce an add-on processing block with all other modules remaining the same as that in an OFDM system. Two alternate transmission methods, other than the mapper, their pros and cons and their complexity with respect to the entire transmitter is presented.

At the receiver end, the matched filter first reconstructs the FTN symbols which are then passed on to the iterative decoder. The inner decoder is specific to decoding of FTN modulated signals and consists of a successive interference canceler that cleans up the two dimensional interference introduced with FTN signaling. The decoding is aided by the outer decoder which corresponds to an error correcting convolutional code decoder. The receiver performance in AWGN channels is evaluated through simulation of the the entire transceiver system. The system is then modeled to operate with finite wordlength precision. The wordlengths are determined so as to maintain a balance between the performance degradation and size of arithmetic units. The system is also evaluated to determine the individual values of T_Δ and F_Δ so as to minimize hardware complexity as well as induced interference (due to FTN signaling). Finally, for transmission of data with fixed block sizes, the actual achievable gains from the FTN system are presented.

Chapter 3

FTN signaling in fading channels

Modern wireless receivers do in reality encounter a different kind of communication channel other than an AWGN channel, that is a fading channel. This is due to the multiple paths, through reflection, refraction or by direct propagation, the transmitted radio waves take before arriving at the receiver. The fading channel, also referred to as frequency selective channels, presents varying amplitudes and phases for different sub-carriers used in a broadband wireless system. Further, due to the mobility of the transmitter, the receiver, or both, the frequency selective channel can vary over time. The frequency and time varying properties of the channel are to be known or evaluated at the receiver in order to recover the transmitted data. The channel values at different sub-carriers are generally referred to as channel coefficients and estimation of these coefficients is called channel estimation [ESvdB+98,Mol05]. Modeling the channel behavior and its properties is referred to as channel modeling. Channel estimation, modeling and measurement are themselves extensively researched topics [Mol05, K̊09] and is not the focus of research in this work. A simplistic approach has been assumed in this work, while more realistic channel models or measured channels that may be used to evaluate the receiver will be a part of future work.

The effect of the time varying frequency selective channel can be compensated with the help of a channel equalizer. In order to do so, the channel parameters at different sub-carriers at a particular time instance has to be known. The study and performance evaluation of FTN systems in frequency selective channels form an important aspect as wireless terminals encounter

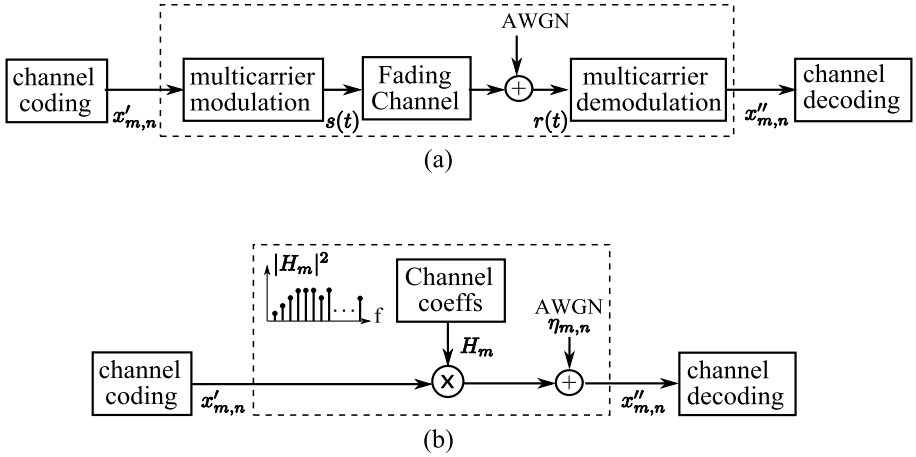

Figure 3.1: Block diagram of a multicarrier system in a fading channel (a) time domain representation, (b) equivalent frequency domain representation.

such channels most often in practice. However, FTN signaling in fading channels has not been studied extensively. Only recently, in [CKRD10] there has been some work on FTN signaling in fading channels and the work is mainly on channel equalization and detection. The work in [CKRD10] uses two static multipath channel realizations with a line of sight component for evaluation of the receiver.

3.1 System model

A general block diagram of the transmit-receive of information in the presence of a fading channel is shown in Figure 3.1(a). If $r(t)$ is the received signal, then the inputs to the channel decoder ($x''_{m,n}$) will be

$$x''_{m,n} \triangleq \int r(t)\, \psi^*_{m,n}(t)\, dt, \tag{3.1}$$

where $\psi_{m,n}(t)$ refer to the IOTA orthogonal basis. In this work, we have restricted ourselves to the system model that uses a frequency domain representation of the channel (Figure 3.1(b)) such that Eqn. (3.1) can be written

as

$$x''_{m,n} = x'_{m,n} \cdot H_m + \eta_{m,n}, \tag{3.2}$$

where H_m represents the channel coefficient at sub-carrier m. The frequency domain model presented in Figure 3.1(b) (and in Eqn. (3.2)) is a simplified model of the channel and the multicarrier modulation operations involved. It has been shown in [LFAB95,LGA01] that IOTA based multicarrier modulation is indeed capable of operating in fading environments without the use cyclic prefix[1]. However, this may not be true for all fading channel conditions, especially when the delay spread or the frequency spread is large. We assume that the channel parameters from the realizations in this work remain well within the limits and work in an orthogonal fashion so that the model presented in Eqn. (3.2) holds true. The extension to fading channel presented in this chapter is a step forward in evaluating FTN signaling for fading environments. A more elaborate study that takes into account the large Doppler and delay spreads is required in future research.

Furthermore, the realized channels are assumed to satisfy the following assumptions:

- The channels are realized by random generation of multipath components having an exponentially decaying power delay profile where each multipath component is assumed to be IID Gaussian with zero mean.
- The number of multipath components can vary depending on the environment in which the receiver is operating, however, it is limited to 6 paths in this work.
- The time variance of the channel is assumed to be static for the duration of the block of information that is received. This implies that the entire received block of information underwent the same type of frequency distortion. This is generally referred to as block fading.

A baseband equivalent fading channel, in this work, refers to the channel realizations satisfying the aforementioned criteria.

[1] The channel models used in [LGA01] are a subset of those specified in the ETSI standard TR 101 112.

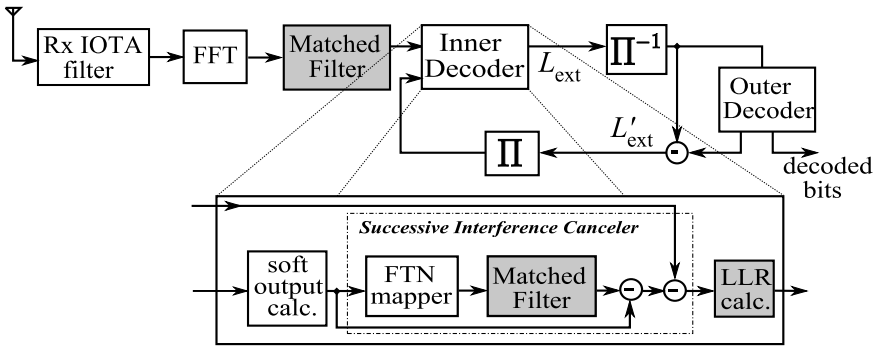

Figure 3.2: Block diagram of the FTN receiver (blocks highlighted in grey process signals taking into account the fading channel).

3.2 Receiver processing in presence of fading

The processing blocks within the receiver essentially remain the same as previously presented for an AWGN channel (Figure 2.8) but with few modifications. In the presence of a frequency selective channel, the receiver processing must now account for the effects of the channel. This is carried out in the MF and parts of the inner decoder in the receiver that follows the FFT. Figure 3.2 presents the simplified block diagram of the receiver for the fading channel, with blocks highlighted in grey being those that need to be altered in order to handle the signals received over a fading channel. The matched filtering and the LLR calculation for the fading channel will be elaborated in detail in the coming sections.

Channel coefficients at orthogonal and FTN sub-carrier positions:

The received FTN modulated block is affected by the fading channel whose coefficients are given by H_k. This is visualized in Figure 3.3 that shows a block of received projections on the orthogonal basis as •s in the time-frequency grid. The channel coefficients H_k correspond to those at orthogonal sub-carrier positions. However, after matched filtering, the reconstructed FTN symbols require channel coefficients at FTN sub-carrier (non-orthogonal) positions, i.e., H_k^{FTN} for the calculation of LLRs in the inner decoder. The channel coefficients at FTN sub-carrier positions (H_k^{FTN}) and those at orthogonal sub-carrier positions (H_k) are related as

$$H^{\text{FTN}}_{\frac{k}{F_\Delta}} = H_k, \qquad 1 \leq k \leq N. \tag{3.3}$$

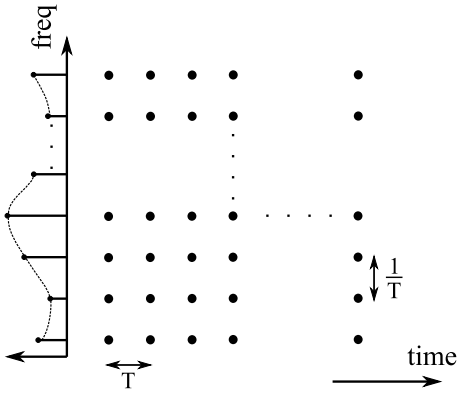

Figure 3.3: Effect of fading on a transmitted information block.

The channel coefficients H_k^{FTN} are not known but can be calculated from H_k. Figure 3.4 illustrates the two scenarios where $F_\Delta = 1$ and $F_\Delta < 1$ and their requirements for calculating H_k^{FTN} from H_k. With the choice of $F_\Delta = 1$, calculation of H_k^{FTN} does not arise. In the case of $F_\Delta < 1$ the channel coefficients H_k^{FTN} are to be calculated from H_k, possibly through interpolation. In our case, with $F_\Delta = 1$, the need for calculating H_k^{FTN} is avoided and can be used interchangeably during the calculations involving the channel coefficients. However, for the sake of clarity, the convention used here is for the general case of $F_\Delta < 1$.

3.2.1 Matched filtering with equalization

The principle behind the MF is to reconstruct the FTN symbols by extracting the energy that was projected onto a set of sub-carriers and time instances by the FTN mapper. When the FTN receiver was evaluated in an AWGN channel, the channel over the transmitted bandwidth was assumed to be flat and without any variable attenuation over sub-carriers. This led to the simple reconstruction equation defining the MF shown in Eqn (2.23) with channel coefficients as unity over all the sub-carriers and with AWGN. Now, in the the presence of fading, the frequency selective channel would have affected the sub-carriers differently. From Figure 3.1(b) (and Eqn. (3.2)), the input to the MF will now be

$$x''_{m,n} = H_m \; x'_{m,n} + \eta_{m,n}. \tag{3.4}$$

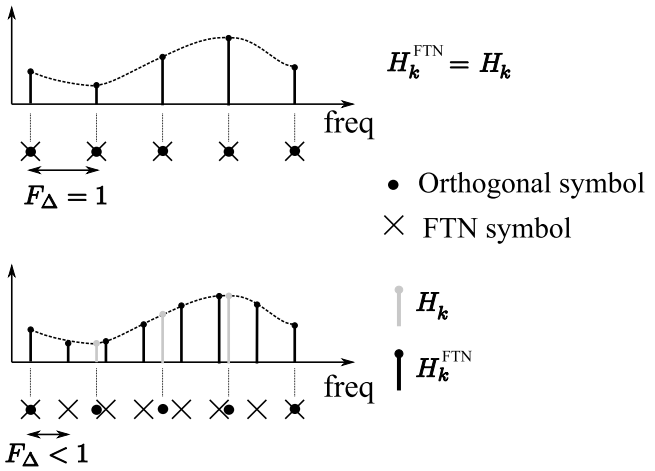

Figure 3.4: Evaluating channel coefficients at FTN sub-carrier positions using those at orthogonal sub-carrier positions.

The reconstruction of the FTN symbols by the MF from the received projections taking into account the frequency selective channel will thus be

$$\bar{x}_{k,\ell} = \sum_{m,n} H_m^* \; C_{k,\ell,m,n} \;\; (x''_{m,n}), \tag{3.5}$$

where * is the conjugation operator on the channel coefficients H_m. This is equivalent to performing equalization to compensate for the frequency selective channel by taking only a proportional amount of energy from each sub-carrier of the channel depending on its strength. It can be noted that setting $H_m = 1$ in Eqn. (3.5) results in an MF for an AWGN channel, previously shown in Eqn. (2.23). Substituting Eqn. (3.4) in (3.5) we get the reconstructed FTN symbols as

$$\bar{x}_{k,\ell} = \sum_{m,n} H_m^* C_{k,\ell,m,n} \;\; \left(H_m \; x'_{m,n} + \eta_{m,n}\right). \tag{3.6}$$

Since the FTN mapper does not change the way the symbols are processed, the equation describing the mapper from Eqn. (2.9) can be readily used to substitute for $x'_{m,n}$ in Eqn. (3.6) to give

$$\bar{x}_{k,\ell} = \sum_{m,n} H_m^* \; C_{k,\ell,m,n} \left(H_m \sum_{k_p,\ell_q} \; x_{k_p,\ell_q} \; C_{k_p,\ell_q,m,n} \right) + \eta'_{k,\ell}, \tag{3.7}$$

where $\eta'_{k,\ell}$ is the colored noise due to the MF operation and is given by

$$\eta'_{k,\ell} = \sum_{m,n} H^*_m \; C_{k,\ell,m,n} \; \eta_{m,n}.$$

At a certain sub-carrier $k = k_1$ and time instance $\ell = \ell_1$, Eqn (3.7) becomes

$$\bar{x}_{k_1,\ell_1} = \sum_{m,n} H^*_m \; C_{k_1,\ell_1,m,n} \cdot \Bigg(\underbrace{H_m \; x_{k_1,\ell_1} \; C_{k_1,\ell_1,m,n}}_{\text{signal component at } k_1,\ell_1} + \underbrace{\sum_{(k_p,\ell_q)\neq(k_1,\ell_1)} H_m \; x_{k_p,\ell_q} \; C_{k_p,\ell_q,m,n}}_{\text{interference at index } k_1,\ell_1} \Bigg) + \eta'_{k_1,\ell_1}. \tag{3.8}$$

Eqn. (3.8) shows that the reconstructed FTN symbol consists of a signal component and noise plus an interference component, similar to the case of an AWGN channel in Eqn (2.25). As before, these reconstructed symbols are iteratively decoded and the noise plus interference component are cleaned up over the iterations. Introducing the modified MF into the SIC results in it being capable of handling the fading channel.

3.2.2 LLR calculation

From Eqn (3.6) we know that the interference canceled symbols during iterative decoding consist of both the signal component as well as the noise plus interference components. In order to calculate the LLRs, the variance of the noise plus interference component is to be found. In general, the variance of the received symbols can be split in a similar way as previously shown in Eqn (3.8) as,

$$\sigma^2(\text{int. canc. symbols}) = \sigma^2(\text{signal component}) + \sigma^2(\text{noise} + \text{interference}).$$

Thus, the variance of the noise plus interference component can be evaluated as

$$\sigma^2(\text{noise} + \text{interference}) = \sigma^2(\text{int. canc. symbols}) - \sigma^2\text{var}(\text{signal component}).$$

This is mathematically denoted as

$$\sigma^2_{(N+I)} = \sigma^2_{(I_c)} - \sigma^2_{(S)}.$$

The variance $\sigma^2_{(N+I)}$, together with the interference canceled symbols, is used in the calculation of LLRs as

$$\text{LLR(int. canc. symbol)} = \frac{2 \times \text{int. canc. symbol}}{\sigma^2_{(N+I)}}. \tag{3.9}$$

The variance used in the LLR calculation, unlike that in an AWGN channel, now varies on a sub-carrier basis due to the frequency selective channel. This brings up the requirement to evaluate the variance at each sub-carrier and is described in the following section.

Variance per sub-carrier

The interference canceled symbols ($\check{x}_{k,\ell}$) obtained as the outputs from the SIC are used in the calculation of LLRs. In the first iteration, with no SIC, the LLRs are evaluated directly on the reconstructed FTN symbols[2]. The number of FTN symbols in the received information block are $N_{\text{FTN}} \times M_{\text{FTN}}$, where we remind the reader that the number of time instances per sub-carrier is M_{FTN}. The variance of the interference canceled symbols at a particular sub-carrier k is defined as

$$\sigma^2_{(I_c)_k} = \frac{1}{M_{\text{FTN}}} \sum_{\ell=1}^{M_{\text{FTN}}} |\check{x}_{k,\ell}|^2. \tag{3.10}$$

The variance of the signal component at each sub-carrier will be

$$\sigma^2_{(S)_k} = \frac{1}{M_{\text{FTN}}} \sum_{\ell=1}^{M_{\text{FTN}}} \sum_{m,n} |C_{k,\ell,m,n}\, x_{k,\ell}\, H_k^{\text{FTN}}|^2, \tag{3.11}$$

here the term $|C_{k,\ell,m,n}\, x_{k,\ell}|^2$ refers to the symbol energy placed at orthogonal symbol (m,n) stemming from FTN symbol at (k,ℓ). H_k^{FTN} is the channel coefficient at sub-carrier k. Since we use offset-QPSK, the FTN modulated symbols are ± 1 and the above equation simplifies to

$$\sigma^2_{(S)_k} = \frac{1}{M_{\text{FTN}}} \sum_{\ell=1:M_{\text{FTN}}} \sum_{m,n} |C_{k,\ell,m,n} H_k^{\text{FTN}}|^2. \tag{3.12}$$

The projection coefficients $C_{k,\ell,m,n}$ are time varying, i.e., they are not independent of the indices k,ℓ. Hence projections from all time instances of the

[2]The reconstructed FTN symbols are on the non-orthogonal grid. Hence the channel coefficients that are to be used for LLR calculation are H_k^{FTN}

information block has to be considered in the calculation. However, they have been approximated by considering coefficients from only one particular time instance, ℓ_1. Hence, Eqn (3.12) can be approximated as

$$\sigma^2_{(S)_k} \approx \sum_{m,n} |C_{k,\ell_1,m,n} H_k^{\mathrm{FTN}}|^2. \tag{3.13}$$

Further, the FTN symbol being projected to $N_f(=3)$ orthogonal sub-carriers, the corresponding channel strengths has to be considered during the evaluation of the variance. For any sub-carrier k, the variance of the signal component considering the channel coefficients will actually be

$$\begin{aligned} \sigma^2_{(S)_k} = &\sum_{m,n} |C_{k,\ell_1,m,n}\, H_{k-1}|^2 \\ &+ \sum_{m,n} |C_{k,\ell_1,m,n}\, H_k|^2 \\ &+ \sum_{m,n} |C_{k,\ell_1,m,n}\, H_{k+1}|^2. \end{aligned} \tag{3.14}$$

Since the channel is assumed to be slowly varying, the coefficients across $N_f = 3$ sub-carriers can be assumed to be constant and equal to H_k^{FTN}, i.e.,

$$H_{k-1} \approx H_k \approx H_{k+1}. \tag{3.15}$$

With the above approximation, Eqn (3.14) becomes

$$\sigma^2_{(S)_k} = |H_k|^2 \sum_{m,n} |C_{k,\ell_1,m,n}|^2. \tag{3.16}$$

Since the total transmitted energy of the projected FTN symbol is very close to 1 (i.e., $\sum_{m,n} |C_{k,\ell,m,n}|^2 \approx 1$), Eqn. (3.16) can be further simplified as

$$\sigma^2_{(S)_k} = |H_k^{\mathrm{FTN}}|^2. \tag{3.17}$$

The approximation in Eqn. (3.13) which only considers a single time instance does not affect the accuracy of the variance being calculated. This is because, had all the time instances been considered, the variance equation would be similar to that in Eqn. (3.16), i.e.,

$$\sigma^2_{(S)_k} = |H_k^{\mathrm{FTN}}|^2 \frac{1}{M_{\mathrm{FTN}}} \sum_{\ell} \sum_{m,n} |C_{k,\ell,m,n}|^2, \tag{3.18}$$

and the term $\frac{1}{M_{\text{FTN}}} \sum_{\ell} |C_{k,\ell,m,n}|^2$ would equal 1 and hence resulted in the same value for the variance as that in Eqn. (3.17). Thus the variance σ^2_{N+I} on a per sub-carrier basis can be calculated as

$$\begin{aligned} \sigma^2_{(N+I)_k} &= \sigma^2_{(I_c)_k} - \sigma^2_{(S)_k} \\ &= \frac{1}{M_{\text{FTN}}} \sum_{\ell=1}^{M_{\text{FTN}}} |\check{x}_{k,\ell}|^2 - |H_k^{\text{FTN}}|^2. \end{aligned} \tag{3.19}$$

Noise power per sub-carrier

Eqn. (3.19) cannot always be used in the LLR calculation since the calculated $\sigma^2_{(N+I)}$ for a particular sub-carrier k may turn out to be negative and hence is an invalid value for the variance. In such cases, the noise power on sub-carrier k is chosen over the estimated variance and is calculated as

$$\sigma^2_{(\text{noise})_k} = \frac{N_0}{2} \frac{1}{M_{\text{FTN}}} \sum_{\ell,m,n} |C_{k,\ell,m,n}\ H_k^{\text{FTN}}|^2 \tag{3.20}$$

$$\approx \frac{N_0}{2} |H_k^{\text{FTN}}|^2 \frac{1}{M_{\text{FTN}}} \sum_{\ell,m,n} |C_{k,\ell,m,n}|^2. \tag{3.21}$$

Here again, the term $\frac{1}{M_{\text{FTN}}} \sum_{\ell,m,n} |C_{k,\ell,m,n}|^2 \approx 1$ in Eqn. (3.21) (c.f. Eqn. (3.18)). Hence the noise power per sub-carrier will be

$$\sigma^2_{(\text{noise})_k} = \frac{N_0}{2} |H_k^{\text{FTN}}|^2. \tag{3.22}$$

LLR calculation

In order that the variance at a particular sub-carrier to be non-negative, it is chosen as

$$\sigma_k^2 = \max\left(\sigma^2_{(N+I)_k},\ \sigma^2_{(\text{noise})_k} \right), \tag{3.23}$$

and the LLR is calculated as

$$L_{\text{ext}}(\check{x}_{k,\ell}) = \frac{2\,\check{x}_{k,\ell}}{\sigma_k^2}. \tag{3.24}$$

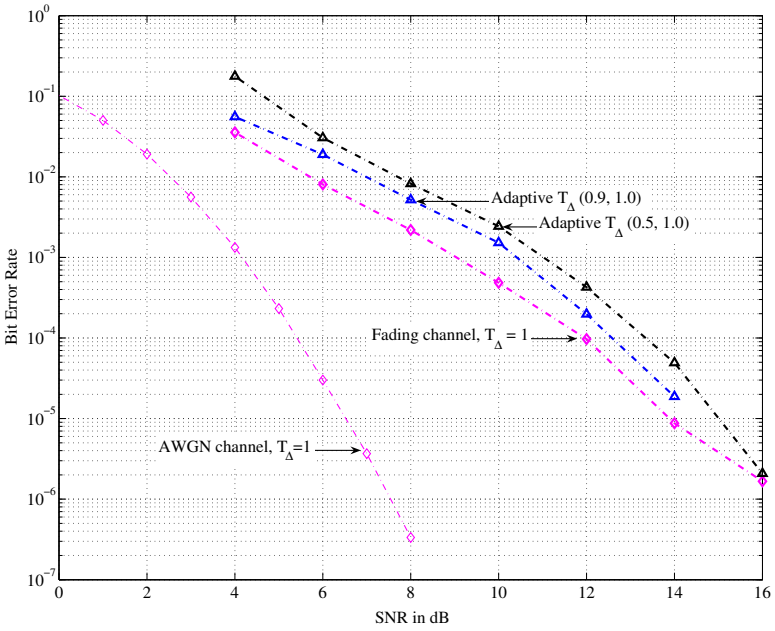

Figure 3.5: BER performance of FTN decoder in the presence of a fading channel.

3.2.3 Results

Figure 3.5 shows the performance of the FTN receiver for a fading channel using the MF and LLR calculation discussed in the above sections. Simulations are carried out by dividing the entire bandwidth of operation into several sub-bands and applying FTN signaling ($T_\Delta = \{0.5, 0.9\}$) in sub-bands that are good (channel strength better than 70% of the average strength) while orthogonal signaling is used in the rest of the sub-bands. The receiver performance is shown relative to the benchmark of the FTN decoder in an AWGN and fading channel corresponding to an orthogonal system (represented as $T_\Delta = 1$ in Figure 3.5). One curve corresponds to the configuration which uses $T_\Delta = 0.5$ and orthogonal signaling, while the other curve uses $T_\Delta = 0.9$ in place of 0.5. It is to be noted that this is heuristic approach and does not consider some system constraints such as the total available transmit power. Instead all sub-bands irrespective

of the channel condition are used for transmission of data. As a result the amount of power used for transmission vary over time. While this is a heuristic approach, it has been presented here to demonstrate the receive processing of FTN signals in fading channels. A more practical and methodical approach of using FTN signaling adaptively is discussed in the following section.

3.3 Adaptive FTN signaling

In the presence of fading, certain frequency bands or time intervals of the channel gets bad. The channel variations can be used opportunistically to transmit information by using different values of T_Δ and adapting to the channel conditions. The choice of T_Δ to be used is based on certain parameters and knowing the state of the channel at transmitter through some form of feedback from the receiver. It is a well known fact that when the channel conditions are good, the decoding or the performance is better and this has been the basis for many algorithms employing variable rate or adaptive modulation schemes [Cav72, GC97]. In this section, we present an approach that takes into account information about the channel condition and employs FTN signaling to maximize the data rate. A similar approach for a MIMO system is dealt in [Rus07], where FTN signaling is employed on antennas with good eigenmodes to improve data rate.

Many modern wireless systems operate with a bandwidth consisting of several hundred sub-carriers. The entire bandwidth of operation is first divided into sub-bands and FTN signaling is used in good parts of the sub-band in order to maximize the data rate. The transmitter is assumed to have information about the sub-bands in terms of the minimum strength of the channel in that sub-band. If the operational bandwidth of the multicarrier system is assumed to consist of N sub-carriers, it is divided into N_{sb} sub-bands consisting of equal number of sub-carriers (N_{sc}) in each sub-band. N, N_{sb} and N_{sc} are related as

$$N_{\text{sb}} = \frac{N}{N_{\text{sc}}}.$$

The channel state defined as a threshold (Th_j) of each sub-band (N_{sb_j}) is determined as

$$\text{Th}_j = \min(|H_{1,j}|, |H_{2,j}| \ldots |H_{N_{\text{sc}},j}|), \qquad 1 \leq j \leq N_{\text{sb}}, \tag{3.25}$$

where $H_{1,2\ldots,N_{sc}}$ are the channel coefficients in sub-band j.

3.3.1 Maximizing data rate with FTN signaling

If R_b is the bit rate that is to be maximized, R_c the coding rate, $R_{s,j}$ the symbol rate in sub-band j and M_{mod} the modulation order or number of bits per symbol; the achievable bit rate from the N_{sb} sub-bands will be

$$R_b = \sum_{j=1}^{N_{\text{sb}}} R_c M_{\text{mod}} R_{s,j}. \tag{3.26}$$

The chosen modulation scheme being OQAM, $M_{\text{mod}} = 2$. The symbol rate in sub-band j is dependent on the time period T of the symbols as well as the FTN signaling rate T_Δ in sub-band j (denoted as $T_{\Delta,j}$), hence

$$R_{s,j} = \frac{1}{T_{\Delta,j}\, T}.$$

If orthogonal signaling is used, $T_{\Delta,j} = 1$. Substituting for $R_{s,j}$ and M_{mod} in Eqn. (3.26) we get

$$R_b = \sum_{j=1}^{N_{\text{sb}}} \frac{2R_c}{T_{\Delta,j}\, T}. \tag{3.27}$$

If P_{Tx} is the total power at our disposal and it is allocated into the sub-bands such that

$$P_{\text{Tx}} = \sum_{j=1}^{N_{\text{sb}}} p_j, \qquad p_j \geq 0$$

where p_j is fraction of the power in sub-band j. The received signal to noise ratio in the j^{th} sub-band becomes

$$\left(\frac{E_b}{N_0}\right)_j = \frac{\frac{p_j}{R_b}\text{Th}_j^2}{N_0}, \tag{3.28}$$

Substituting for R_b and simplifying,

$$\left(\frac{E_b}{N_0}\right)_j = \frac{p_j \text{Th}_j^2 T_{\Delta,j}\; T}{2R_c N_0}. \tag{3.29}$$

If we assume that in order to achieve a target BER of P_e it is required to operate at an SNR of XdB. Then the constraint on the SNR $\left(\frac{E_b}{N_0}\right)_j$ is that it

should be larger than the SNR that achieves the target BER, i.e.,

$$\left(\frac{E_b}{N_0}\right)_j > X, \qquad 1 \le j \le N_{\text{sb}} \tag{3.30}$$

then the sub-band j may be operated at T_Δ such that

$$T_\Delta = \{0.5, 0.6, 0.7, 0.9\}. \tag{3.31}$$

Though there is a choice for $T_{\Delta,j}$, in order to make the best use of the channel the system needs to be operated at the smallest achievable T_Δ, 0.5 in our case.

The bit rate R_b in Eqn. (3.26) can be maximized by

$$\begin{aligned} &R_b = \max_{\tau, p_j} \sum_{j=1}^{N_{\text{sb}}} \frac{2R_c}{T_{\Delta,j} T}, \\ &\text{such that} \\ &\frac{p_j \text{Th}_j^2 T_{\Delta,j}\, T}{2R_c N_0} = X, \qquad 1 \le j \le N_{\text{sb}} \\ &T_{\Delta,j} = T_\Delta, \\ &\sum_{j=1}^{N_{\text{sb}}} p_j \le 1. \end{aligned} \tag{3.32}$$

substituting $\left(\frac{1}{T_{\Delta,j}\, T}\right) = R_{s,j}$ in Eqn. (3.32), we get

$$\begin{aligned} &R_b = \max_{p_j, R_{s,j}} 2R_c \sum_{j=1}^{N_{\text{sb}}} R_{s,j}, \\ &\text{such that} \\ &\frac{p_j \text{Th}_j^2}{2R_c N_0 R_{s,j}} = X, \qquad \Rightarrow R_{s,j} = \frac{p_j \text{Th}_j^2}{2R_c N_0 X}, \qquad 1 \le j \le N_{\text{sb}} \\ &\frac{1}{T R_{s,j}} = T_\Delta, \qquad \Rightarrow 0 \le R_{s,j} \le \frac{1}{T_\Delta\, T}, \qquad 1 \le j \le N_{\text{sb}} \\ &\sum_{j=1}^{N_{\text{sb}}} p_j \le 1. \end{aligned} \tag{3.33}$$

Reformulating in terms of p_j

$$R_b = \max_{p_j} \frac{1}{N_0 X} \sum_{j=1}^{N_{\text{sb}}} p_j \text{Th}_j^2,$$

such that

$$0 \leq p_j \leq \frac{2R_c N_0 X}{T_\Delta \, T \text{Th}_j^2}, \qquad 1 \leq j \leq N_{\text{sb}} \tag{3.34}$$

$$\sum_{j=1}^{N_{\text{sb}}} p_j \leq 1.$$

If N_{sb} sub-bands have thresholds $\text{Th}_1, \text{Th}_2 \ldots \text{Th}_{N_{\text{sb}}}$, the power allocation is carried out by first rearranging the channel thresholds such that $\text{Th}_1 \geq \text{Th}_2 \geq \text{Th}_3 \cdots \geq \text{Th}_{N_{\text{sb}}}$. Then the available power P_{Tx} is allocated into the sub-bands starting from that sub-band which is best followed by the next best and so on. From Eqn. (3.34), the amount of power required in the first sub-band will be

$$p_1 = \min \left(\frac{2R_c N_0 X}{T_\Delta \text{Th}_1^2 T}, 1 \right), \tag{3.35}$$

followed by the allocation of the remaining power $(P_{\text{Tx}} - p_1)$ into the subsequent $(N_{\text{sb}} - 1)$ sub-bands as

$$p_j = \min \left(\frac{2R_c N_0 X}{T_\Delta \text{Th}_j^2 T}, 1 - \sum_{k=1}^{j-1} p_k \right), \qquad 2 \leq k \leq N_{\text{sb}}. \tag{3.36}$$

3.3.2 Results

The power allocation scheme in Eqn. (3.35) and (3.36) was used to evaluate the achievable bit rates when operating the system at different T_Δ. It was shown earlier in Section 2.6 that a BER of 10^{-5} was achievable at an SNR of 6 dB with the use of the outer $(7, 5)$ convolutional code. The target BER was set at 10^{-5} and hence the system operated at an SNR of $X = 6$ dB. The coding rate $R_c = 1/2$ due to the use of rate 1/2 $(7, 5)$ convolutional code. The system is assumed to consist of $N = 1000$ sub-carriers with $N_{\text{sb}} = 20$ sub-bands.

The channel realization was according to the parameters described in Section 3.1. Random realizations of 10^6 channels were generated and the power allocation algorithm was run in order to evaluate the achievable bit rate. Figure 3.6 shows the achievable number of bits per T seconds per sub-band as a function of SNR when $T_\Delta = \{0.5, 0.6, 1.0\}$ are used.

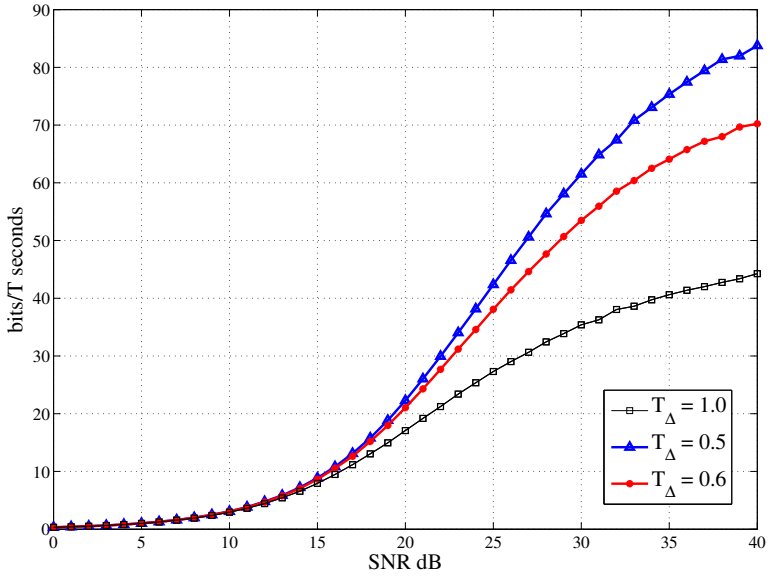

Figure 3.6: Achievable bit rates per sub-band in a multicarrier system with N sub-carriers divided into N_{sb} sub-bands and using power allocation to maximize data rate using different levels of FTN signaling.

3.4 Summary

This chapter has explored FTN signaling in frequency selective channels. First, the MF and the inner decoder is modified in order to account for the fading. The calculation of variance of noise and interference for LLR in the frequency selective channel is deduced from the AWGN assumption presented in earlier chapter.

A power allocation approach in order to improve the data rate is presented. This is achieved by dividing the operating bandwidth into a number of sub-bands. The sub-bands experiencing better channel conditions are prioritized and allocated with power using the power allocation scheme making the best use of the available power.

Chapter 4

FTN Transmitter: Hardware Architecture and Implementation

This chapter discusses the hardware architecture and implementation of the mapper previously proposed in Section 2.1. It was shown that the mapper approach can realize both orthogonal and FTN signaling without making major changes in the transmitter. In other words, excluding the mapper and the IOTA filter, the components in the transmitter correspond to an orthogonal (OFDM) system, c.f Figure 2.5. The hardware implementation of the IOTA filter and its complexity with respect to the FTN system will be discussed in Chapter 7. This implies that the overhead in the transmitter due to FTN signaling can be tackled by an efficient implementation of the mapper.

4.1 Look-Up Table based architecture

Figure 4.1 shows a generic block diagram of the FTN mapper detailing the representation of several FTN symbols onto a single sub-carrier, contributing with different weights. In general, N such blocks are required to produce the outputs for a multicarrier system with N sub-carriers. The mapper in Figure 4.1 was defined in Eqn (2.9) as

$$x'_{m,n} = \sum_{p,q} x_{k_p,\ell_q} \cdot C_{k_p,\ell_q,m,n}. \tag{4.1}$$

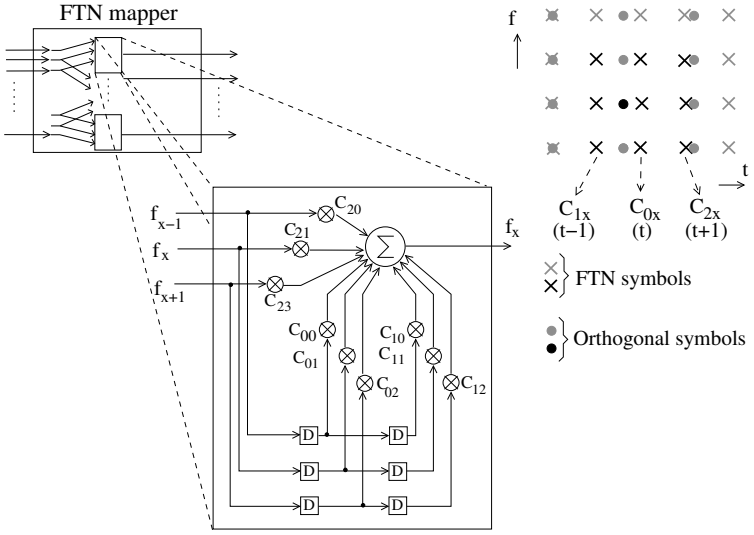

Figure 4.1: Generic architecture of the FTN mapper (detail:for single sub-carrier).

Since each block requires $N_t \times N_f$ operations, the complexity or the number of operations carried out in the mapper during each time instance will be $(N \times N_t \times N_f)$ multiplications and $N \times ((N_t \times N_f) - 1)$ additions. The incoming FTN symbols are OQAM modulated with possible input values being ± 1 only. This implies that the projection coefficients are to be either added/subtracted and requires no multiplications. Thus the computational complexity will be only of $O\{N \times ((N_t \times N_f) - 1)\}$ additions. However, when higher order modulations are considered, the incoming symbols have to be multiplied with the projection coefficients before accumulation.

The mapper can be simplified with a look-up table (LUT) based approach with pre-calculated coefficients. From Figure 2.15, it can be seen that the FTN and the orthogonal symbols overlap at an interval. If T_Δ and F_Δ are non-recurring decimals, then the overlapping interval is finite resulting in the projection coefficients having a repetitive property in time and frequency. For example, consider $T_\Delta = 0.6$; if both the FTN and orthogonal symbols start at time instance 0, they again overlap at time instances 3 and 6. The projection coefficients corresponding to the FTN symbol at instance 6 is same as that at instance 0. The coefficients at instances 0 and 3 do not match as they have opposite phase due to the choice of OQAM modulation; however instances 3 and 9 will have the same projection coefficients. This results in 10 FTN

Table 4.1: Repetition rates of FTN symbols ($F_\Delta = 1.0, F_{\text{rep}} = 2$) and $N_t = N_f = 3$.

spacing in time (T_Δ)	repetition in time (T_{rep})	LUT size ($N_t \times N_f \times T_{\text{rep}} \times F_{\text{rep}}$)
0.4	10	180
0.5	4	72
0.6	10	180
0.7	20	360
0.9	20	360
1.0	2	36

symbols that have unique projection coefficients along time ($0, 0.6, \ldots, 5.4$ or $3, 3.6, \ldots, 8.4$ or $6, 6.6, \ldots$ and so on). This separation between FTN symbols after which the projection coefficients repeat is referred to as repetition rate. A similar situation happens along sub-carriers. Table 4.1 gives a list of repetition rates for different T_Δ. The frequency spacing has been fixed to $F_\Delta = 1.0$ and hence $F_{\text{rep}} = 2$ for all configurations.

If T_{rep} and F_{rep} are the repetition rates of the FTN projection coefficients in time and sub-carriers, then

$$\text{LUT size} = N_t \times N_f \times T_{\text{rep}} \times F_{\text{rep}}. \tag{4.2}$$

By storing the above set of values, all possible FTN symbols for that configuration of T_Δ and F_Δ can be represented. In other words, the lookup table size becomes independent of the information blocksize of the system when T_Δ and F_Δ are chosen as non-recurring decimals. As the total number of precalculated projection coefficients are small they do not have significant impact on complexity overhead.

4.1.1 Operating at sub-optimal points

This section provides the basis for choosing sub-optimal operating points for a particular $T_\Delta F_\Delta$ earlier presented in Section 2.4. The sub-optimal choice helps to minimize the size of the look-up tables (LUTs) that store the projection coefficients. Consider the $T_\Delta F_\Delta = 0.4$ curve in Figure 2.10 that has least interference at $T_\Delta = 0.55$ and consequently $F_\Delta = 0.4/0.55 = 0.7272\ldots$. Ideally, this is the operating point of choice for minimal interference with FTN signaling. However, choosing this configuration will result in, $T_{\text{rep}} = 20$ and

$F_{\text{rep}} = \infty$. As can be seen from Eqn. (4.2), T_{rep} and F_{rep} directly factor in the equation determining the LUT size. When operating at the point of least interference, though $F_{\text{rep}} = \infty$ need not have to store infinite values but will now depend on the number of sub-carriers in the system. Therefore,

$$\begin{aligned} \text{LUT size for optimal } T_\Delta F_\Delta &= N_t \times N_f \times T_{\text{rep}} \times N, \\ &= 3 \times 3 \times 20 \times N, \\ &= 180N. \end{aligned} \tag{4.3}$$

However, at the chosen operating point $T_\Delta = 0.4$ and $F_\Delta = 1$, from Table 4.1,

$$\begin{aligned} \text{LUT size for sub}-\text{optimal } T_\Delta F_\Delta &= 3 \times 3 \times 10 \times 2, \\ &= 180. \end{aligned} \tag{4.4}$$

This value is independent of the number of sub-carriers or block size chosen for the FTN system. In this configuration, for a system with just 100 sub-carriers, the savings in LUT size in the chosen operating point when compared to the optimal point is a factor of 100. Though savings might not be this drastic in every case, there is still a significant complexity reduction in the LUT sizes at the chosen operating points. It can be concluded that having $F_\Delta = 1$ is advantageous despite the marginally higher interference.

4.2 Implementation

The LUT based generic architecture for the FTN mapper is shown in Figure 4.2. Since the coefficients representing the ISI/ICI on each orthogonal sub-carrier are already stored in LUTs and the incoming modulated symbols being ± 1, the coefficients simply need to be accumulated before being transmitted. This is performed by the accumulator array following the LUT. The number of accumulators required would be $N_t \times N_f$, in our case it is 9 since $N_t = N_f = 3$ and is sufficient when using IOTA basis. The datapath controller accesses the correct value from the LUT by keeping track of the time instance and the sub-carrier frequency of the incoming FTN symbols. Because of the repetition rates, the address generation is simplified to modulo counters in T_{rep} and F_{rep}. Each LUT location stores a word consisting of $N_t \times N_f$ values and hence the depth of the LUT will be $T_{\text{rep}} \times F_{\text{rep}}$ from Eqn. (4.2). In each access, the LUT provides $N_t \times N_f$ values pertaining to ISI for N_t time instances and ICI across N_f sub-carriers caused by one FTN symbol. These are accumulated along with the effects from previous FTN symbols. For this a buffer that can hold values associated with N sub-carriers for N_t (=3) time instances is used in the output stage.

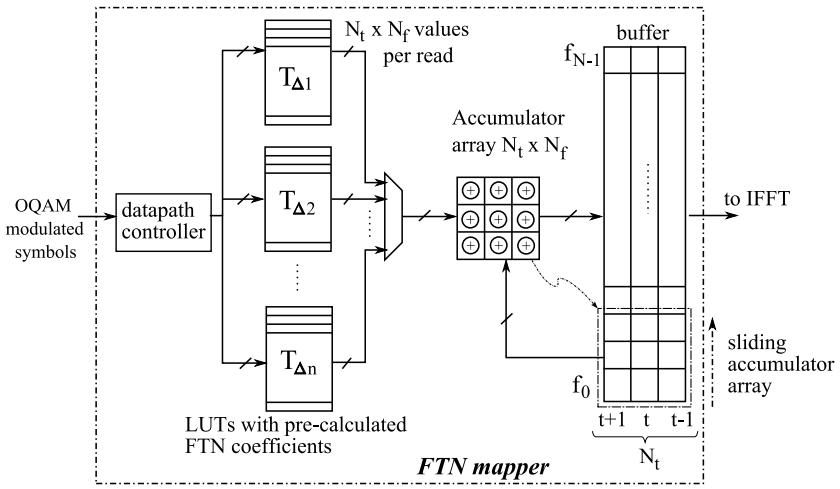

Figure 4.2: Generic Look-Up Table based FTN mapper architecture.

The accumulator array and the buffer (see Figure 4.2) repeatedly collects the ISI/ICI effects from the FTN symbols. Once new FTN symbols appear that no longer affect the oldest N orthogonal sub-carriers stored in the buffer, they are passed on to the IFFT block for multicarrier modulation. The remaining ones are realigned to accommodate the calculations with new incoming values. The LUTs are implemented using ROMs, while the implementation of the buffer can be done using a register bank or a RAM. Register based implementation tend to be faster as any of the values can be readily accessed. However, for systems with large number of sub-carriers, the area due to the register bank can be overwhelming. If RAMs are used for buffers, the speed has to be traded off for area. The following sub-sections brings out the pros and cons of two approaches that are evaluated for mapper implementation.

4.2.1 Register based implementation

The register based FTN mapper is shown in Figure 4.3 and uses a bank of registers as the buffer to store the partial results of the mapping. The advantage of using registers in the buffer is that the calculation corresponding to each incoming FTN symbol can be completed within a single clock cycle. The LUT has been implemented as a combinatorial logic and a small delay occurs when reading out the values. It can be seen from the figure that every incoming OQAM modulated FTN symbol looks-up $N_t \times N_f$ values from the look-up table. These LUT outputs are to be accumulated with the corresponding set

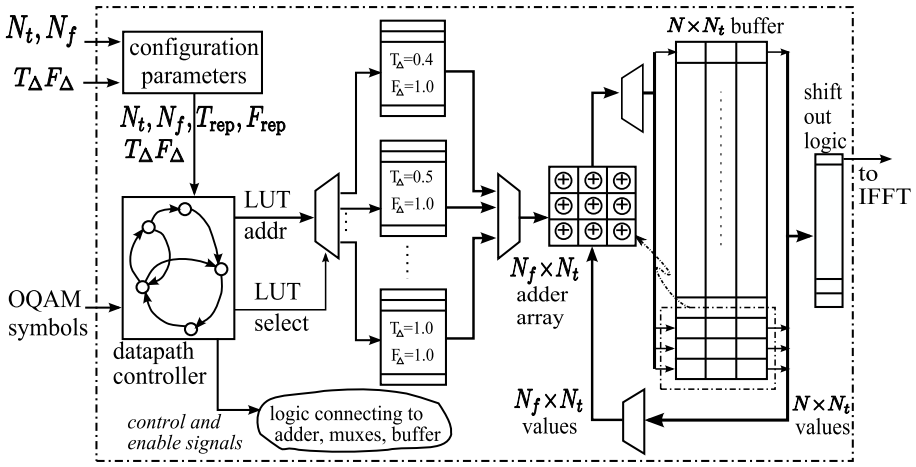

Figure 4.3: Register based FTN mapper architecture.

of previously stored results from the buffer and stored back into the buffer. Summing the values from the LUT with corresponding locations in the buffer is also combinatorial in nature. Hence it is only required to choose the registers whose values are available at their outputs through a MUX, and added along with the LUT values using an adder array. Thus, the result at the output of the adder array is ready by the following clock edge to be stored back into the registers. The writing back of the result can be done by an enable signal on the appropriate set of registers in the buffer. The timing diagram in Figure 4.4 shows the read, calculate and write back operations happening within one clock cycle.

Though this approach seems to be the preferred solution due to its speedy operation, it has to be noted that the multiplexer (MUX) and de-multiplexer (de-MUX) between the adder array and bank of registers depend on N, N_t and N_f. In general, a $(N \times N_t) : (N_f \times N_t)$ multiplexer from the register outputs to the adder array and an equal size demultiplexer from the adder output to the register input will be required. If $N = 128$, $N_f = N_t = 3$ then a $384 : 9$ line multiplexer and $9 : 384$ line demultiplexer will be required. This results in quite a large amount of combinatorial resources requiring significant amount of routing. Further, the buffer implemented using registers also tend to be a dominant resource consuming part of the entire mapper. Hence this approach is not attractive for implementation especially when $N > 64$.

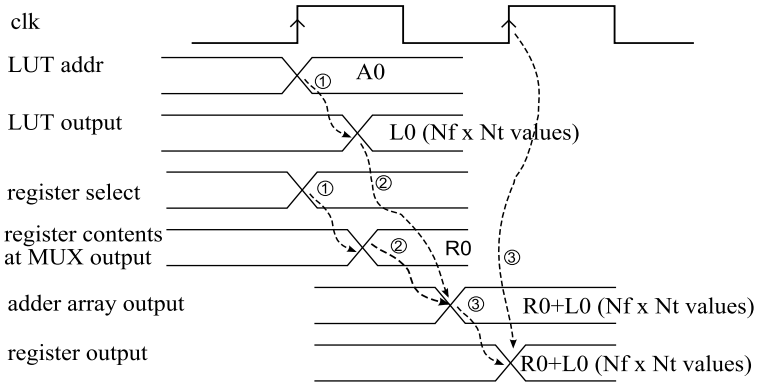

Figure 4.4: Timing diagram of register based architecture showing LUT read, buffer access, calculation and write back.

4.2.2 RAM based implementation

A considerable amount of resources consumed in buffers, multiplexers and routing in the register based FTN mapper can be significantly avoided by the use of RAMs with some trade off in speed. This is due to the fact that only one location can be accessed at a time unlike the register based approach in which any number can be read by just tapping their outputs. The RAM based architecture is shown in Figure 4.5 where the RAMs are used as buffers. Each column in the original buffer (Figure 4.3) is replaced by a RAM module with the same depth (N) as the original buffer. Each RAM now stores one value corresponding to a time instance and hence requires 3 RAMs, when $N_t = 3$. One reason for using different RAMs is because, by doing so, 3 values can be read out simultaneously. This can also be done by having one wide RAM that holds 3 values in one location. However, when it comes to shifting out the result, values corresponding to one time instance are to be shifted out. During this time the RAM cannot be accessed to carry on with the calculations as it will be used to shift out the result. Until then the calculation for the new FTN symbols has to be stalled. Since it is only a part of the entire contents of the RAM that corresponds to the output, the remaining ones are to be written back after re-formatting the values resulting in a lot of data transfer operations which is inefficient when it comes to power consumption. In summary, the use of a single RAM as a buffer will lead to 'process and wait' situations for the FTN mapper and a lot of data rearrangement.

In order to have a pipelined operation between the calculation stages, 3 separate RAMs, one corresponding to each column, is instantiated so that data

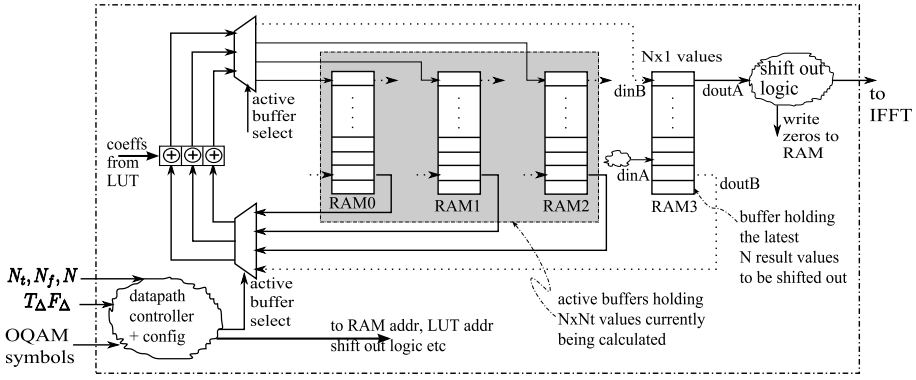

Figure 4.5: RAM based FTN mapper architecture.

can be read out and written into the RAMs simultaneously. Further, to make the shifting out the result and calculation of newer incoming data to happen in parallel an extra RAM is instantiated. Figure 4.5 illustrates that at any given time 3 RAMs are involved with the datapath controller to perform calculations while the fourth one contains the result from the most recent calculation that needs to be passed on to the IFFT block. The RAM holding the result is handled by the 'shift-out-logic' to read out the data, clear the contents and prepare it to be used for the next set of calculations by the datapath controller. The RAMs involved in the calculations are active in a cyclic fashion and only the outputs corresponding to the active RAMs are selected and read/written by the datapath controller, while the fourth RAM is left to the control of shift-out-logic. In Figure 4.5, the greyed out portion shows the currently used RAMs by the datapath controller to perform calculations and their input output ports are connected to the adder array (shown by solid lines). The fourth RAM is not involved in the calculation of the outputs and hence logically disconnected from the datapath controller (shown by dashed lines).

When it comes to arithmetic units, now only 3 (N_t) adders are sufficient in the adder array as only 3 values can be read out from the RAMs in a particular clock cycle. Thus, the LUT contents are also modified to provide only 3 values at a time. This means that the datapath controller is slightly modified, i.e. the projection of every FTN symbol happens in 3 steps because of the limitation in access to the RAM. Further, the one clock cycle read latency constrains the calculation to a total of 9 clock cycles per FTN symbol (3 memory locations $\times$ 3 clock cycles per location) and this can be improved by the use of a pipelined adder. The two scenarios are shown by a timing diagram in Figure 4.6, where

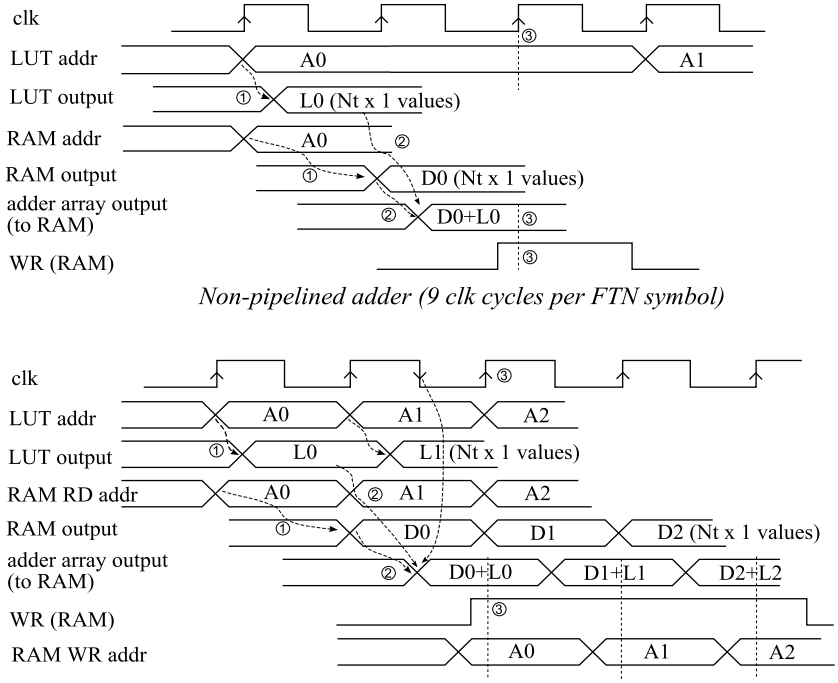

Figure 4.6: Timing diagram for RAM based FTN mapper without and with pipelined adder.

the first case is without a pipelined adder requiring 3 clock cycles per memory location (and hence a total of 9 clock cycles per FTN symbol) while the second one uses a pipelined adder and the total number of clock cycles reduces to 5. The pipelined adder version of the RAM based approach is chosen for implementation as the rate of calculation can be almost doubled with an additional pipeline stage at the adder outputs. Further, the RAM can also be effectively utilized as it can be accessed to read/write during every cycle of operation, while idle states exist in the non-pipelined version.

Table 4.2: FTN mapper area comparison on Xilinx FPGA.

Architecture	Logic Cells	RAMs (128×16)	No. of adders	18×18 Multipliers
Register based	16 979	0	9	-
RAM based	1 009	4	3	3
128 pt. IFFT core	1 712	7	-	9

4.3 Results

The two flavors of the FTN mapper architecture has been implemented and evaluated for both FPGA and ASIC. The results refer to a multicarrier system with 128 sub-carriers and $N_t = N_f = 3$ projection points for the FTN mapper.

4.3.1 FPGA Implementation

Table. 4.2 provides the resource usage numbers for the design targeted to a Xilinx FPGA (Virtex-II Pro) [Xilb]. Apart from the resource utilization of the FTN mapper, the table also provides resource usage for a 128-point IFFT used for MCM in the transmitter of an OFDM system. This IFFT core is generated using the Xilinx CORE generator™ [Xila]. The IFFT core is presented for comparison with the FTN mapper as it is the one with significant complexity in the transmitter. It can be seen that the logic cells of the RAM based FTN mapper is less than 60% of the IFFT block. Also, the block RAMs used and the actual arithmetic resources (adder/multipliers) are also about half of that of the IFFT. The FTN mapper has been successfully tested and verified on the FPGA. The outputs from the FPGA are compared with the reference MATLAB model. In the mapper implementation, the LUT uses 12 bits for representing the values with 11 bits for the fractional part. The output wordlength is 16 bits wide, while the input is just 1 bit corresponding to the OQAM modulated symbols which are ± 1. The register based architecture is quite expensive in terms of the resource usage and has to be avoided when the sub-carriers of the OFDM system is larger than 64. It also has to be noted that in a system having large number of sub-carriers, even the IFFT block is not a direct mapped implementation for the reasons of area requirements. With a time multiplexed design for the MCM block, the preceding FTN mapper can also employ a similar approach saving hardware real estate. The RAM based implementation provides N outputs from the mapper to the IFFT block at regular intervals. The IFFT architecture can be chosen to match the output rate of the FTN mapper to have the data being computed in a pipelined fashion.

Table 4.3: FTN mapper area comparison in 130nm standard CMOS process.

	Register based arch		RAM based arch.	
	Area in μm^2	% age	Area in μm^2	%age
Buffer/RAM	275 052	68.24%	145 323	60.52%
Adder array	10 324	2.56%	5 000	2.08%
Shift out logic	90 936	23.30%	51 016	21.25%
Datapath controller	18 663	4.63%	27 535	11.47%
Look-up Tables	4 976	1.23%	11 105	4.62%
Configuration block	136	0.03%	147	0.06%
Total area	538 179	100.00%	240 126	100.00%

4.3.2 ASIC Synthesis

Table. 4.3 gives the comparison of FTN mapper synthesized in 130 nm standard cell CMOS process [UMC] for the two architectures. The table lists the resources consumed by each block within the FTN mapper and it can be seen that the buffer and the shift-out-logic in the register based version consumes the most area. This is avoided by instantiating RAMs for the buffers which also reduces the area occupied by the shift-out-logic as it now just reads out the result values from one of the RAMs. Figure 4.7 shows the final layout of the RAM based FTN mapper. It is evident from the figure as well as Table 4.3 that the memories consume almost 80% of the area, while the actual logic (LUT, controller, adder array) is significantly small. The implementation reports a speed of 330 MHz in the 130 nm process.

The data path controller has larger area in the RAM based approach because the mapping of one FTN symbol is carried out in 3 steps instead of calculation in one go as in the register based approach. Though the adder units are reduced by a third, the reported area does not confirm this because the adders used in the RAM based implementation is pipelined to improve the speed of calculations. The LUTs in the two architectures are designed in the following ways:

1. Each LUT location in register based approach store 9 values.
2. The LUT locations in RAM based approach hold 3 values and are 3 times larger than the LUT in register based implementation.

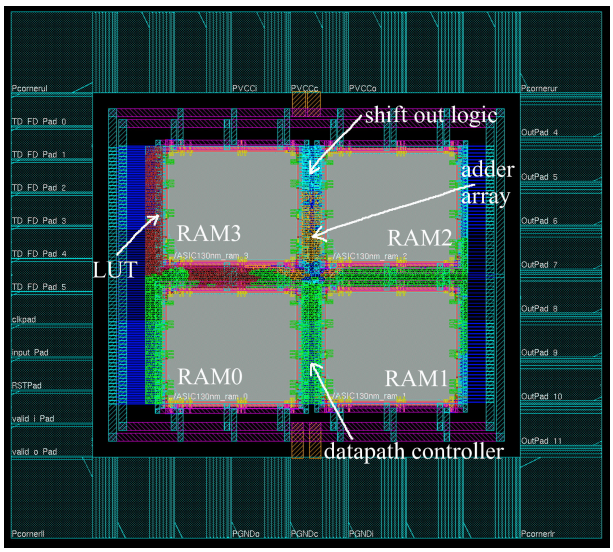

Figure 4.7: Layout of RAM based implementation of the FTN mapper in 130nm CMOS process.

As a result, the address decoder for the LUTs in RAM based architecture tend to have larger overhead than that in the register based counterpart which is shown in the comparison table. Overall, the RAM based approach saves more than 50% of the area when compared to the register based architecture.

4.4 Summary

This chapter has presented a look-up table based hardware architecture for realizing FTN signaling in the transmitter. The FTN system is operated with time and frequency spacings T_Δ and F_Δ at sub-optimal operating points so that the LUT that stores the projection coefficients have a repetition pattern resulting is small sizes of LUTs. Initially a register based implementation was proposed. Though such an implementation is fast, it has a high area overhead. Hence, speed is traded for area by using RAMs instead. The RAM based architecture has been verified on a Xilinx FPGA for its functionality and its complexity is compared with an IFFT implementation. The mapper was also synthesized for a 130nm CMOS process and was found that memories were a dominant factor in the FTN mapper.

Chapter 5

FTN Receiver: Hardware Architecture and Implementation

The design and implementation of baseband processing blocks is one of the key challenges in wireless receivers. Earlier, the transmitter was evaluated to demonstrate that it need not be too complex but that an add-on processing block can realize transmission of data using FTN signaling. A similar approach is undertaken in the implementation of the decoder in the FTN receiver. This chapter discusses the hardware architecture and implementation of parts in the receiver responsible for decoding the received symbols into bits. The receiver proposed in Section 2.3 already did consider re-using the processing blocks to realize different functions hinting at less overhead in the receiver. The following sections detail the hardware architecture of each of the blocks and the motivation to realize so. The proposed hardware architecture primarily focus on the inner decoder as it is specific to FTN signaling. While the outer decoder is a max-Log MAP approximation of the BCJR decoder for the $(7, 5)$ convolutional code.

The architectural description of the processing blocks are organized in the same way as the received symbols are processed in the receiver. The IOTA pulse shaping filter used as a part of multi-carrier demodulation is separately discussed in Chapter 7. The FFT, being one of the most extensively researched topic for efficient and optimized hardware implementation is not considered for the very reason. The hardware architecture of the remaining processing blocks will be discussed before applying any optimizations. A simplified block diagram

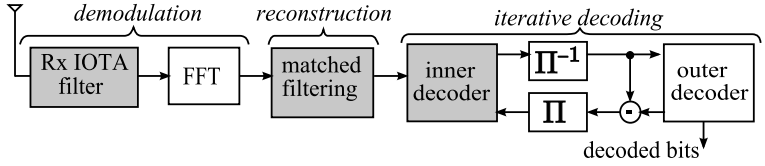

Figure 5.1: Block diagram of the FTN receiver chain.

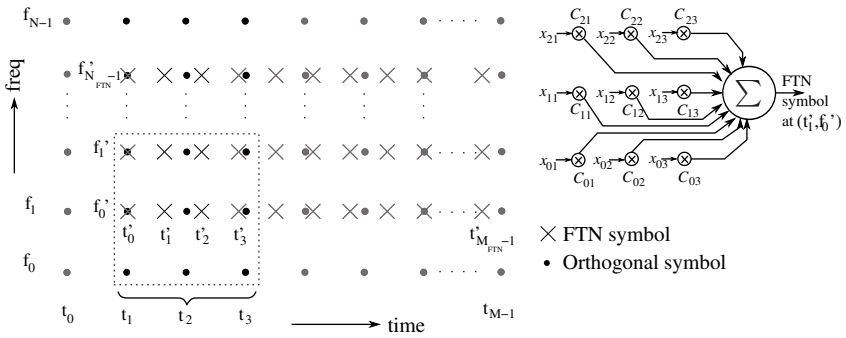

Figure 5.2: Time frequency grid showing the MF operation and computational diagram of Eqn. (2.23).

of the receiver is recollected from the previous chapters and is shown in Figure 5.1. The following sections consist of descriptions of the matched filter, inner decoder with soft output calculation, successive interference canceler, and the LLR calculation followed by a brief description of the outer decoder.

5.1 Matched Filter architecture

The hardware architecture of the matched filter algorithm described in Section 2.3.1 is explained using Figure 5.2. The time instances and sub-carriers for the FTN symbols are denoted as t'_ℓ and f'_k while, $\mathrm{t_n}$ and $\mathrm{f_m}$ denotes indices for orthogonal symbols. In order to reconstruct the FTN symbol at $(\mathrm{t}'_1, \mathrm{f}'_0)$, the orthogonal symbols at time instances $\mathrm{t}_1, \mathrm{t}_2, \mathrm{t}_3$ and sub-carriers $\mathrm{f}_0, \mathrm{f}_1, \mathrm{f}_2$ are required when $N_t \times N_f = 3 \times 3$ is used. The symbols at these orthogonal time instances are denoted as $x_{01}, x_{02} \ldots, x_{23}$ with the corresponding projection coefficients $C_{01}, C_{02} \ldots, C_{23}$. The matched filter operation requires $N_t \times N_f$ multiplications whose outputs are accumulated to obtain the reconstructed FTN symbol.

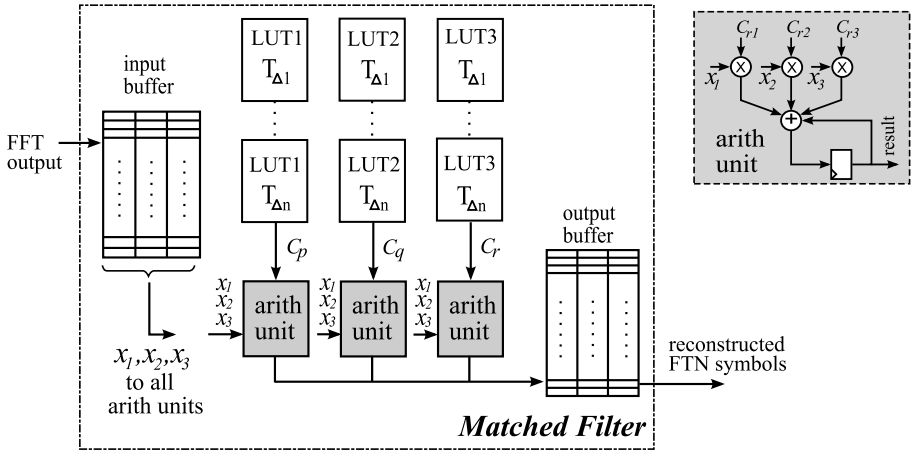

Figure 5.3: Architecture of the matched filter with triplicated LUTs and arithmetic units.

The maximum number of simultaneous FTN symbols that can be calculated when 3 time instances of orthogonal symbols are available varies from $1-3$, depending on T_Δ. Several FTN symbols can be calculated concurrently because each FTN symbol at the transmitter is projected onto the nearest N_f sub-carriers and N_t time instances with respect to the FTN symbol. Hence, with a smaller time spacing there can be several FTN symbols that can be mapped onto the same set of orthogonal basis functions. Of all the time spacings considered in this work, $T_\Delta = 0.4$ has the smallest separation between adjacent FTN symbols, which gives the highest number of FTN time instances that may be calculated simultaneously ($= 3$). Accordingly, 3 arithmetic units and LUTs will be required to calculate the FTN symbols in parallel.

For illustration consider the time-frequency grid in Figure 5.2, when time instances t_0, t_1 and t_2 are available, only 1 of the 3 arithmetic units will be used to compute the output corresponding to FTN time instance t'_0. Similarly, 2 arithmetic units will be required to calculate FTN time instances t'_1 and t'_2 when t_1, t_2 and t_3 are available. In this way, depending on the orthogonal time instances currently being processed as well as time spacing T_Δ, the arithmetic units $1-3$ are enabled accordingly.

The architecture of the matched filter with 3 arithmetic units is shown in Figure 5.3. It consists of 3 buffers indicated as 'input buffer' which stores the demodulated symbols and are read into the arithmetic units for FTN symbol reconstruction. If FTN symbols corresponding to 2 time instances are calcu-

lated simultaneously, then two arithmetic units are enabled and a controller reads out the respective coefficients from the 2 LUTs. The controller handles the enabling of arithmetic units, generation of proper coefficient addresses, and writing into the output buffer. Since RAMs are used as buffers, the reconstruction operation is carried out in 3 cycles, each corresponding to 1 sub-carrier, due to the limitation of accessing several address of the RAM simultaneously. The reconstructed symbols are stored in the output buffer with each column corresponding to a time instance. Again, if 2 time instances are calculated in parallel, 2 columns of output buffer is used to store the results and so on.

The choice of calculating FTN symbols at different time instances concurrently at the cost of replicating the LUTs is motivated as follows. From Figure 5.2, it can be seen that reconstruction of FTN symbols at t'_1 and t'_2 both involve the same set of values that need to be read from the input buffer. This means that the same memory locations should be accessed twice or thrice resulting in larger latency, multiple memory access as well as increased power consumption. The proposed scheme avoids repeated read accesses to a large extent. After every calculation, values corresponding to one orthogonal time instance, t_n, can be discarded from the input buffer. This is important because freeing the input buffer at a constant rate will not stall the preceding blocks. The MF being a component of the SIC in the inner decoder, the proposed approach helps in speeding up the process. It should be noted that due to the repetitive property of projection coefficients (Table 4.1) the sizes of LUTs are not very big (same size as that used in the mapper) and hence their duplication does not have a significant impact on complexity. Furthermore, the overhead due to the replication of arithmetic units is acceptable when compared to the speed up it provides, especially for systems with large number of sub-carriers.

5.2 Inner Decoder architecture

The inner decoder is the component decoder responsible for decoding of FTN modulated symbols. However, it also includes a successive interference canceler that cleans up the interference from the FTN symbols in each iteration. In order to perform SIC, the LLRs from the outer decoder are converted back to soft symbols by the 'soft output calculation' unit. These soft outputs together with the reconstructed FTN symbols from the MF are then used by the SIC for interference cancellation. The 'LLR calculation' unit uses the so obtained symbols to evaluate the LLRs which are passed to the outer decoder. In the strict sense the inner decoder is only the LLR calculation unit, but in this work it collectively refers to the soft output, SIC and LLR calculation units.

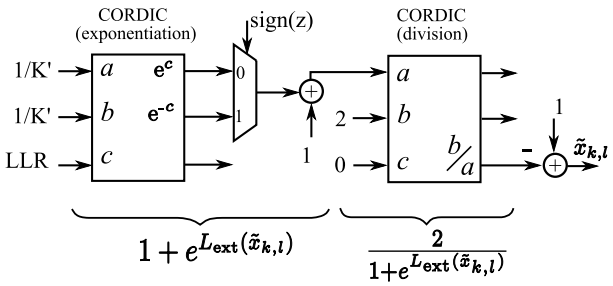

Figure 5.4: Soft output calculation using CORDIC.

5.2.1 Soft Output Calculation

The soft output calculation block reads the LLR values from the interleaver and calculates the soft output value. The equation describing the soft output calculation as (c.f. Eqn. (2.26))

$$\tilde{x}_{k,\ell} = 1 - \frac{2}{1 + e^{L_{\text{ext}}(\tilde{x}_{k,\ell})}}. \tag{5.1}$$

Looking at Eqn. (5.1), we can see that it requires an exponentiation operation and a division operation apart from trivial arithmetic. Exponentiation and division operation can be efficiently implemented using Co-Ordinate Rotation DIgital Computer (CORDIC) [Vol59] and is shown in Figure 5.4. Initially the CORDIC was implemented to evaluate the soft outputs. However, the entire soft output calculation in Eqn. (5.1) was later realized as a look-up table. This is because the input and output wordlength of the soft output calculation block is set at 8 bits (4 integer and 4 fractional bits) during the wordlength evaluation in Section 2.6.1. Further, with the dynamic range of the input LLRs being ± 5, the LUT needs to store only 161 values. On the other hand, if w is the input wordlength, then the number of stages generally required by a CORDIC is also w [Par00]. Therefore, realizing a function using CORDIC would require w $\times$ w CORDIC units. Given that each CORDIC unit consists of 3 adders/subtractors, 2 shifters [Par00], the LUT approach turns out to be very appropriate in terms of area.

5.2.2 SIC using mapper-matched filter cascade

Figure 5.5 shows the proposed hardware architecture of the successive interference canceler. The SIC is carried out by first evaluating the total interference experienced by each FTN symbol. Then it is canceled from the corresponding FTN symbol. While the cancellation is simply a subtraction operation,

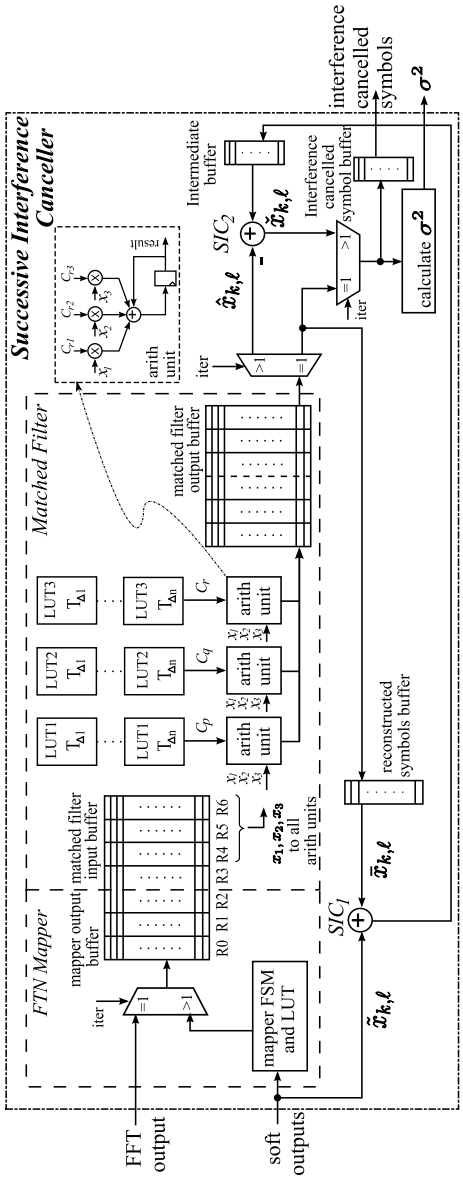

Figure 5.5: Architecture of the successive interference canceler using mapper-matched filter cascade.

estimating the interference experienced by each FTN symbol is more complex. In order to calculate the interference, the mapper (from the transmitter) and the matched filter (from the receiver) are used in conjunction as was described in Section 2.3.2. Apart from reusing the blocks that were used in a different context, reordering of the operations is discussed in the following to carry out a pipelined operation. The reconstructed FTN symbols, $\bar{x}_{k,\ell}$, (in Figure 5.5) produced during the first iteration is stored in the *reconstructed symbols buffer*. During the subsequent iterations the soft outputs, $\tilde{x}_{k,\ell}$, are calculated and provided to the mapper-MF cascade to produce $\hat{x}_{k,\ell}$. Realizing Eqn. (2.32) as is requires that the soft symbols are accessed twice, once to calculate $\hat{x}_{k,\ell}$ and once for subtraction in SIC. This can be avoided by simply reordering the sequence of operations in Eqn. (2.32) as

$$\begin{aligned}\check{x}_{k,\ell} &= \bar{x}_{k,\ell} - \{\hat{x}_{k,\ell} - \tilde{x}_{k,\ell}\} \\ &= \bar{x}_{k,\ell} - \ \hat{x}_{k,\ell} + \tilde{x}_{k,\ell} \\ &= \{\bar{x}_{k,\ell} + \tilde{x}_{k,\ell}\} - \hat{x}_{k,\ell}. \end{aligned} \tag{5.2}$$

The above rearrangement implies that, every time soft outputs ($\tilde{x}_{k,\ell}$) are read into the FTN mapper they are also summed up with the reconstructed FTN symbols ($\bar{x}_{k,\ell}$) and is denoted by SIC_1 in Figure 5.5, and is stored in the *intermediate buffer*. Once the outputs $\hat{x}_{k,\ell}$ are available, the corresponding result from the *intermediate buffer* is read and SIC_2 is performed to produce $\check{x}_{k,\ell}$ which is stored in the *interference canceled symbol buffer*.

The blocks in Figure 5.5 are demarcated to have one-to-one correspondence to the *SIC* in Figure 2.9. The mapper is the same as that implemented for the transmitter [DAK+09] discussed in Section 2.1 (Figure 4.5) while the matched filter is from Figure 5.3. The buffers between mapper and MF in Figure 5.5, are shared in a cyclic fashion. For example, at any time when the mapper is writing to $R2, R3$ and $R4$, the MF uses buffers $R6, R0$ and $R1$. In the following time instance the $R5$ will be taken over by the mapper while the MF uses buffers $R0, R1, R2$. This becomes possible by instantiating 2 duplicate LUTs and arithmetic units leading to a pipelined style of processing at the block level freeing one column of buffer and engaging another. Though at any point of time the mapper writes into 3 columns and the MF reads from 3 columns in the shared memory, 7 memories are instantiated. This is done in order to prepare the buffer freed by the MF before being used by the mapper. Similarly, the buffers at the output of the MF is shared between the MF and the interference cancellation logic. As described in Section 5.1, the MF can produce outputs up to 3 time instances simultaneously. So 6 buffers are instantiated between the MF and SIC logic in order to accommodate the MF to calculate outputs corresponding to a maximum of 3 time instances in succession. Thus while the

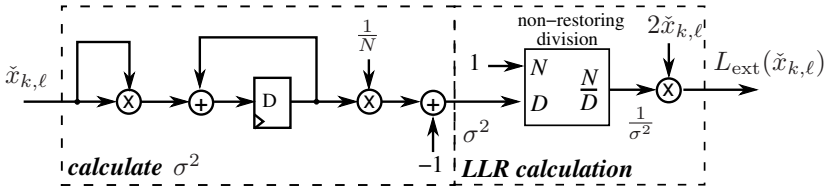

Figure 5.6: Noise variance and LLR calculation.

MF is writing new outputs, the SIC logic clears up the memory by performing interference cancellation on the previously calculated results. The MF and SIC logic alternately control either the first or the last 3 columns of the shared memory denoted as 'matched filter output buffer' in Figure 5.5.

5.2.3 LLR calculation

The LLR is calculated using the variance estimate of the noise + interference (σ^2_{N+I}) and requires interference canceled symbols of the entire transmitted information block to be available beforehand, c.f. Eqn. (2.35). However, calculating σ^2_{N+I} can be carried out in concurrence with the SIC operation. Hence, by the completion of interference cancellation on the entire received block, σ^2_{N+I} will be available for LLR calculation.

The calculation of the variance is realized using 2 multipliers and an accumulator as shown in Figure 5.6 and is a straightforward implementation of Eqn. (2.35). The last stage shows the LLR calculation using the inverse of noise variance. To calculate LLRs using Eqn. (2.34) implies that $N_{\text{FTN}} \times M_{\text{FTN}} (= K)$ division operations are to be carried out, i.e., dividing each interference canceled symbol by the noise variance. This can be avoided by first calculating the inverse of σ^2_{N+I} and then multiplying it with the interference canceled symbols as

$$\begin{aligned} L_{\text{ext}}\{\check{x}_{k,\ell}\} &= \frac{2\check{x}_{k,\ell}}{\sigma^2_{N+I}} \\ &= \frac{1}{\sigma^2_{N+I}} \left(\check{x}_{k,\ell} << 1\right), \end{aligned} \tag{5.3}$$

where $\check{x}_{k,l} << 1$ indicates a left bit shift of $\check{x}_{k,l}$ by 1 position realizing multiplication by 2 at no extra hardware cost. Hence, the actual process of LLR calculation requiring $N_{\text{FTN}} \times M_{\text{FTN}}$ division operations is now reduced to 1 division and $N_{\text{FTN}} \times M_{\text{FTN}}$ multiplications. A multiplication can be carried out in a single clock cycle resulting in speed improvement. The division operation is realized using non-restoring approach and calculates the result in as many

clock cycles as the wordlength of the input. This approach is effective because the division operation is carried out only once per iteration and does not have a significant effect on the processing speed.

5.3 Outer Decoder

The outer decoder uses the BCJR algorithm [BCJR74] for soft decoding of the $(7, 5)$ convolutional code. The log-domain computations of the BCJR results in an efficient implementation popularly called as log-MAP algorithm [RVH95]. A further simplification of the log-MAP algorithm is called the max-log-MAP algorithm and is popularly used for hardware implementations. The log-MAP and max-log-MAP algorithms are very widely researched topic that can be readily found in literature [PS08, LC04].

The max-log-MAP implementation in the FTN decoder[1] uses a single window approach, a straight forward approach for hardware implementation without any optimization considerations for throughput, area or power. The single window approach requires 3 computational units called α, β and γ together with a memory. α and β are called state metric units and γ the branch metric unit. The representation and their names have a established meanings in literature for BCJR, log-MAP and max-log-MAP algorithms. A simplified implementation for the outer decoder is mainly chosen to work together with the inner decoder and demonstrate the hardware functionality of the FTN. As a result, the implementation is not discussed in any further detail as the focus is not the optimization or improvement of the outer decoder.

5.4 Controller for the FTN decoder

A global FSM is the top level controller that handles the flow of data in and out of the chip, and between the decoders. The global FSM is also used to parameterize the number of iterations, T_Δ, F_Δ for FTN signaling. Figure 5.7 shows the state diagram of the global FSM used to configure the FTN parameters prior to decoding and handle the data flow between the component decoders.

The controller is in the 'IDLE' state to begin with. Once a 'start decoding' trigger is received, the decoding parameters (number of iterations, $T_\Delta F_\Delta$) are read in and configured. Then the controller moves into the 'read inputs' state. After reading in the entire blocklength of data that is to be decoded, the

[1]The hardware implementation of the max-log-MAP decoder has been carried out by Master students Lay-Hong Ang, Wee-Guan Lim, Yuan Mengze and Cai Meng.

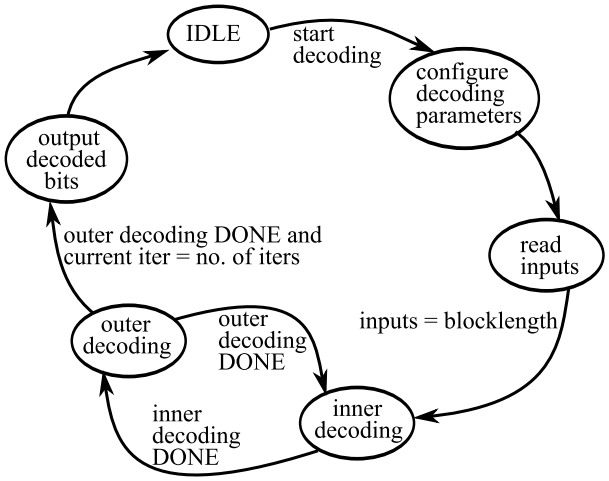

Figure 5.7: State diagram showing the global controller for the FTN decoder.

iterative decoding begins. Once the number of iterations becomes equal to the configured number of iterations, the decoded bits are made available at the output. Finally, the controller goes back to the 'IDLE' state and waits for the next set of data to decode.

5.5 Implementation results

The FTN iterative decoder has been synthesized for 65 nm standard cell digital CMOS process from ST Microelectronics [ST] as well as for Xilinx FPGA [Xilb]. Table 5.1 shows the resource usage of the FTN decoder for both FPGA and ASIC technologies. The FPGA area is given in terms of number of slices occupied on a Xilinx device while the area for ASIC is in μm^2. The table also provides the area usage of the different components of the FTN decoder logic as a percentage of the overall area. From the percentage numbers, it is evident that the logic area match consistently between the ASIC and FPGA implementations. The size of the RAMs, used as buffers, is also indicated in the table.

Table 5.1: Resource utilization for the FTN iterative decoder synthesized for Xilinx FPGA and ST 65nm standard cell CMOS process.

Blocks	Xilinx FPGA (XC2VP30) [Xilb]			ST 65nm CMOS process [ST]				RAM sizes (bits)
	Slices	Logic area (%age)	Block RAMs	Logic (μm^2)	Logic area (%age)	Memory (μm^2)	Memory (%age)	
Inner Decoder	**5015**	**71.6%**	**16**	**109 487**	**68.9%**	**233 979**	**64.1**%	-
- Soft output calc	21	0.3%	-	490	0.3%	-	-	-
- LLR calc	107	1.5%	-	4 048	2.5%	-	-	-
- SIC/FTN Mapper	1666	23.7%	7	28 621	18.0%	65 739	18.0%	$7 \times 128 \times 10$
- SIC/Matched Filter	2928	41.8%	6	64 039	40.3%	36 597	10.0%	$6 \times 128 \times 8$
- SIC/Temp. buffers	293	4.1%	3	12 181	7.6%	131 643	36.0%	$3 \times 2016 \times 8$
Outer Decoder	**734**	**10.4%**	**1**	**20 039**	**12.6%**	**46 260**	**12.6**%	1008×40
Π and Π^{-1}	**1132**	**16.1%**	**2**	**26 094**	**16.4%**	**84 524**	**23.1**%	$2 \times 2016 \times 8$
Global FSM	**120**	**1.7%**	-	**2 767**	**1.7%**	-	-	-
FTN Decoder	***7001***	***100.0%***	***19***	***158 750***	***100.0%***	***360 763***	100.0%	*136 064 bits*

5.5.1 Area

The total synthesized chip area of the FTN decoder for the ST65 nm process is 0.519 mm^2 of which

- 158 750 μm^2 is in the Logic and,
- 360 763 μm^2 in the Memory.

The inner decoder occupies 69%(= 109 487 μm^2 out of 158 750 μm^2) of the overall logic, while the memory occupies about $2/3^{rd}$ of the entire inner decoder. Memory also dominates the overall design taking 69% of the total chip area and the total memory size is about 17 kB. This memory in the FTN decoder is split as:

- 7.75 kB in the Inner decoder.
- 4.92 kB in the Outer decoder.
- 1.96 kB in Interleaver.
- 1.96 kB in De-interleaver.

The FTN mapper and matched filter form significant parts of the inner decoder, while the soft output and LLR calculation blocks together form about 3% of the area. The matched filter implementation has 3 instantiations of arithmetic units, resulting in a 3-parallel system and is the reason for larger area. With a single arithmetic unit, the throughput would be reduced as some matched filtering operations will then need to access the same memory more than once (c.f Section 5.1). In such a case, the number of output buffers in the matched filter can also be reduced to 1. The 3-parallel implementation has been chosen, over the low area choice, to maintain a pipelined operation between the modules as well as to achieve higher throughput. Apart from these, the 3 temporary buffers in the SIC consume considerable portion of the memory area. One buffer is used to store the reconstructed symbols, the other for storing the interference canceled symbols and the third to store partial results during the SIC operation. The global controller occupies about 2% of the logic area and the interleaver and de-interleaver together constitute about 16% of the total area. The low area resource for the outer decoder is due to the design choice of using a single window implementation for the max-log-MAP decoder.

5.5.2 Speed and Throughput

The throughput of the FTN decoder is evaluated by calculating the number of clock cycles (cc) required to process each received symbol, the operating frequency and number of iterations. The reported critical path for the synthesized

design was 2.92 ns implying a peak operating clock frequency of 333 MHz (3 ns clock period). The inner decoder takes 6cc per coded bit with the FTN mapper consuming 5cc [DAK+09]. The outer decoder takes 1cc to process a pair of coded bits, thus requiring an average of 6.5cc per transmitted bit by the FTN decoder. For the considered block size of 2016 bits processed for 8 iterations, the FTN decoder provides a raw data throughput of 6.4 Mbps, resulting in an information throughput of 3.2 Mbps (a rate 1/2 code is used). This figure is without the use of any optimization techniques for high throughput rates, such as parallel window implementation. Even with this baseline implementation the data throughput matches the downlink packet access speeds referred to in the 3G standard [DPS08, 3GP10].

5.5.3 Power consumption

Power is estimated by performing gate-level simulations on the synthesized netlist and logging the toggle information. The power consumption was estimated to be 44.7 mW for the entire FTN decoder running at 200 MHz. The memories dominate the power consumption with 36.7 mW of the total reported power only due to the memories. Dual port RAMs have been used within the FTN decoder as they help in achieving better throughput by providing read and write accesses at the same time. However, the 65nm process technology being used does not allow for low-leakage dual port RAMs and these RAMs are one of the sources of relatively high power consumption.

On a global scale, optimization for power or throughput can be carried out in the following ways. While decoding the FTN modulated signals, either the outer or the inner decoder is processing the information block while the other is idle. This can be exploited to reduce power consumption by powering down the decoder that is inactive. On the other hand, throughput can be improved by processing 2 different blocks of information alternately between the inner and outer decoders. However, this requires doubling the interleaver and de-interleaver memories to buffer the 2 different information blocks.

5.6 Hardware overhead with FTN signaling

One of the primary goals in this work has been on evaluating algorithm hardware tradeoffs for inclusion of FTN signaling in multicarrier systems. The transmitter and receiver architectures proposed have the possibility of including FTN as add-on processing while retaining the conventional multicarrier transceiver chain. Also, reuse of processing blocks have helped in keeping the hardware overhead under check. In order to evaluate the overhead involved in

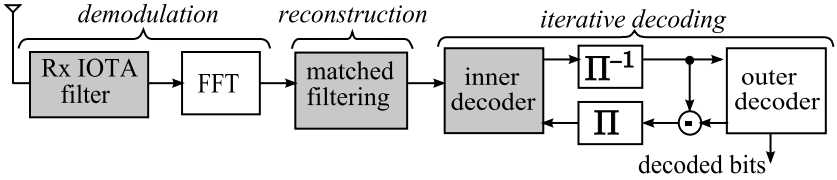

Figure 5.8: Simplified block diagram of the FTN receiver.

FTN signaling, the decoder proposed in this work has to be compared with a reference decoder providing a similar performance. However, there exist no hardware implementations or architectures proposed for these systems and research is limited to theoretical studies [LG03, LL04, BFC09, HZ09, MS10]. Most recently a hardware architecture has been proposed in [WPID11] but for a transmitter. The mapper based transmission scheme proposed in this work has already been compared with [WPID11] in Section 2.2. At the receiver end, the reference decoder considered for comparison are conventional iterative decoders as they have comparable performance as that presented in this work.

Figure 5.8 presents a simplified block diagram of the FTN receiver proposed in Section 2.3. Looking at the block diagram , the interleaver and de-interleaver are parts of any iterative decoding scheme and will not have any influence on the complexity overhead. Furthermore, the outer decoder in the proposed setup corresponds to a max-log-MAP implementation of the BCJR algorithm for the $(7, 5)$ convolutional code. This can be considered as a standard implementation if the same convolutional code were used in both systems. Hence, the comparison of complexity overhead is predominantly on the inner decoder and the matched filter as they are specific to decoding of the FTN modulated signals. From the results presented in Section 5.5 it can be seen that the inner decoder is about 5 times as large as the outer decoder. However, the processing blocks within the inner decoder such as the matched filter is also used outside the iterative decoding to perform FTN symbol reconstruction in the receiver. Similar is the case for FTN mapper, which can be time multiplexed for use in the transmitter apart from being used as a component in SIC. The other dominating fraction in the inner decoder are the buffers used within the SIC. Since memories are a dominating fraction, memory optimizations results in better silicon usage and lower power consumption.

The other overhead in receiver processing is the IOTA filter used in conjunction with FFT to perform multicarrier demodulation. A hardware mapped implementation on the filter was carried out using the same CMOS process as that used for the decoder implementation [ST]. The input and output wordlengths used were those evaluated from fixed point simulations i.e., 8 bits for the in-

puts and the filter coefficients. The filter consisted of 1024 taps implemented as 128 filter banks each consisting of 8 taps. Each filter bank in the IOTA filter occupied about 1690 μm^2, resulting in a total size of 0.208 mm^2 for the overall filter consisting of 128 filter banks. Of this 80% of the filter area is due to the registers in the delay chain. Though the overall area is about 40% of the FTN decoder, the IOTA filter provides two fold advantage of complexity reduction in mapper [DRAO09] and matched filter as well as the elimination of the cyclic prefix [LGA01] that take about $20 - 25\%$ of the OFDM symbol. Furthermore, the Rx IOTA filter needs to run only for a fraction of the time to process a received block of information when compared to the FTN decoder that processes the same information block for several iterations. Hence, the IOTA filter occupying 40% of the area of FTN decoder does not translate to the same percentage of power consumption, further substantiating the processing overhead. Though a fully parallel implementation of such a large filter is not done in practice, it is only for illustration that this comparison is made. A detailed section describing an efficient hardware implementation of the IOTA filter is discussed in Chapter 7.

5.7 Architectural optimizations to reduce area and power

In the work presented so far, the following optimization were already considered at various levels. Algorithmic level optimizations was carried out while designing the FTN transceiver system model. Most of the choices were made to keep the number of operations per FTN symbol low. In the transmitter, choice of IOTA based multicarrier modulation resulted in requiring fewer operations per transmitted FTN symbol [DRAO09], Section 2.1.1. At the receiver, it was shown that operating the FTN system slightly away from the optimal time-frequency spacing ($T_\Delta F_\Delta$) can result in reduced complexity [DRO10], Section 2.4.

On the implementation front, a look-up table for soft output calculation was used by exploiting the fact that the log-likelihood ratios (LLRs) can be restricted to a small dynamic range. This avoided exponentiation and division operations that were needed to calculate soft outputs. Reordering the sequence of subtraction operations during SIC resulted in avoiding duplicate memory access. In the LLR calculation block, N division operations were reduced to 1 division and N multiplications by using the inverse of the noise variance. The matched filter is time shared between FTN symbol reconstruction and SIC (Figure 5.5) providing area savings.

From the results of the baseline implementation of the FTN decoder (Section 5.5), it was found that memory was a dominant source of power consumption and silicon resources. Hence further improvements to reduce the memory requirements in the FTN decoder is proposed. The focus of the memory optimization is primarily on the inner decoder. Since there exist innumerable architectural optimizations for a max-log-MAP implementation, i.e., the outer decoder, it has not been considered. Furthermore, memory within the inner decoder in the baseline implementation accounted for 65% of the overall memory area and 80% of the total estimated power (35mW out of 44mW) hence considered for further optimizations.

5.7.1 Memory optimization

In this section, the memory architecture in the baseline implementation is briefly described followed by buffers chosen for optimization. A simplified architecture of the SIC block is presented in Figure 5.9. The SIC unit in the inner decoder uses 3 buffers (RAMs) of 2kB each, large enough to hold the information block being decoded. The *reconstructed symbols buffer* stores the output of the MF operation from the first iteration which are used until the last iteration of the decoding cycle, hence memory reduction of this buffer is not possible. The *intermediate buffer* stores the result of SIC_1 (c.f. Figure 5.9) at the same time as the soft symbols are read into the FTN mapper. The output of the FTN mapper-MF sequence is used together with corresponding values from the *intermediate buffer* for SIC_2 (c.f. Figure 5.9) resulting in interference canceled symbols. They are stored in the *interference canceled symbol buffer* as well as passed on to the noise variance calculation block to estimate the variance of the noise+interference (σ^2_{N+I}, hereby simply referred to as σ^2). Once all symbols in the received information block have been processed by the inner decoder, σ^2 is used to calculate the LLRs (dashed line in Figure 5.9). The following subsections detail the optimizations of the *intermediate* and *interference canceled symbol buffers*. The blocks shaded in grey in Figure 5.9 indicate that they will be discarded by the proposed optimization process. Several small buffers (of size 128×10) are used within the FTN mapper and the MF. Those are not considered for optimization as they do not provide any significant reduction in the current architecture.

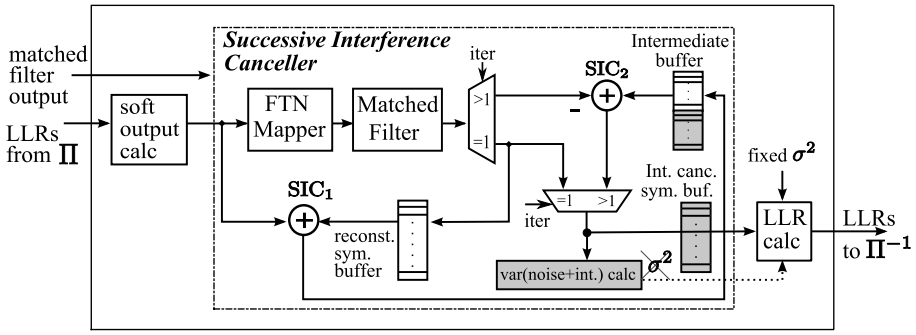

Figure 5.9: Simplified architecture of the SIC.

5.7.2 Intermediate buffer optimization

Problem description

The *intermediate buffer* is used as a FIFO to store the result of SIC_1 (c.f Figure 5.9) before being consumed during SIC_2. The memory size in the pre-optimized design is as large as the information blocksize, i.e. 2kB. This buffer size can be reduced as stored values are emptied at a certain rate before the buffer is completely filled. The minimum size memory requirement of the *intermediate buffer* is understood by analyzing the data flow within the SIC. The FTN symbols span 126 of the 128 sub-carriers while 2 of them are reserved at the end [DRO10]. In time, they span 16 time instances and can be visualized from the generalized depiction in Figure 5.10, again with FTN symbols as $\times$ and the orthogonal basis functions as $\bullet$. t'_{n} and f'_{n} correspond to time and frequency index of FTN symbols, while t_{n} and f_{n} correspond to the indices of orthogonal basis functions. In the SIC, data is processed in sets of 126 symbols corresponding to one time instance (t'_{n}) of the received information block and is hereby referred to as a 'data slot'.

In each iteration, the interference is estimated by passing the soft symbols through the FTN mapper and MF. The FTN mapper projects the incoming FTN symbol ($\times$ at t'_{n}) onto 3 orthogonal basis functions each in time and frequency ($\bullet s$ at $t_{\mathrm{n}}, f_{\mathrm{n}}$), the number 3 is derived in [DRAO09]. After that the MF uses the projected values from the respective orthogonal time instances to reconstruct the FTN symbols. For the MF to begin computations, the FTN mapper should have output data corresponding to time instance t_0, t_1, t_2 (3 time instances in general). As an example, for the FTN configuration shown in Figure 5.10, the FTN mapper projects the symbols at t'_0 onto t_0, t_1, t_2; symbols

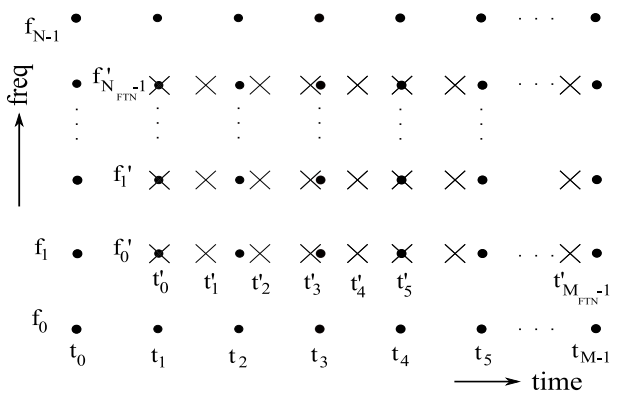

Figure 5.10: Generalized time-frequency grid of FTN and Orthogonal symbols.

at t'_4 onto t_2, t_3, t_4; t'_5 onto t_3, t_4, t_5 and so on [DRAO09]. This implies that the MF will start calculation only after symbols at time instance t'_4 has been completed by the FTN mapper. Now, the SIC_2 operation that consumes data from the *intermediate buffer* requires outputs from the MF which will in turn be available after the FTN mapper has processed time instances $t'_0 - t'_4$. Hence, for the example in Figure 5.10, the *intermediate buffer* should be large enough to hold the result of SIC_1 corresponding to time 5 instances until it has been read out for SIC_2.

Proposed solution

In order to determine the size of the *intermediate buffer*, the maximum waiting time for the MF has to be calculated. This corresponds to the FTN system that is operating with the lowest spacing between the symbols, i.e. $T_\Delta = 0.4$ in our case. By finding a memory size required for the worst case FTN configuration, all other FTN configurations i.e., $T_\Delta = \{0.5, 0.6, 0.7, 0.9\}$ will operate without memory contention problems.

This memory requirement evaluation, determined when $T_\Delta = 0.4$, is presented with the help of Figure 5.11. The figure shows the actual timing of data and control signals, from the point when the result of SIC_1 is written into the *intermediate buffer* until it is consumed during SIC_2. From Figure 5.11, it can be seen that the MF starts operating when 8 of the 16 data slots have been processed by the FTN mapper. Hence the memory size is initially estimated to store as many values i.e., 8 data slots or 126×8 values = 1008 bytes. <1> in Figure 5.11 represents the 16 data slots input into the FTN mapper. Simulta-

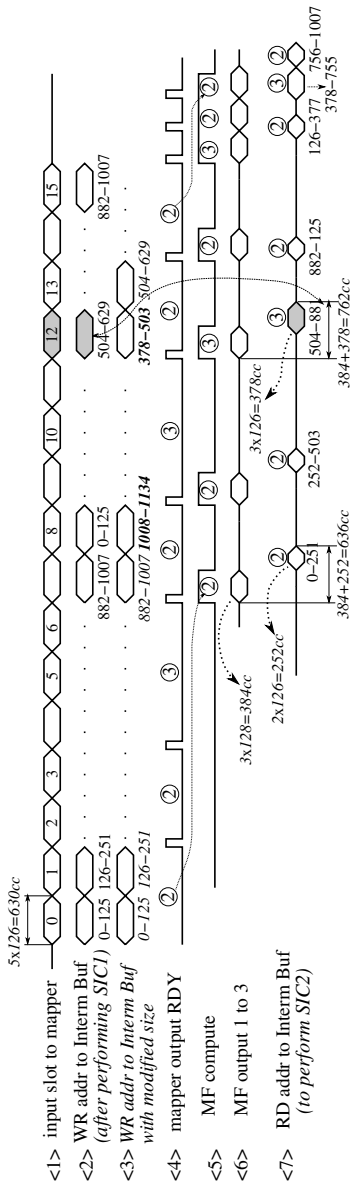

Figure 5.11: Diagram showing timing between mapper-MF and accesses to Intermediate buffer with FTN system operating at $T_\Delta = 0.4$.

neously, these values are used in SIC_1 and result written into the *intermediate buffer* (of size 1008 bytes) and their addresses are shown in <2>. <4> shows the output ready signal from the FTN mapper each time it has completed writing into the memory corresponding to orthogonal time instance t_{n}. Numbers within the circles denote how many data slots have been processed, which is also passed on to the MF. The enabling of MF computations is shown by <5>.

The MF has 3-parallel instantiations of the arithmetic units in order to reduce duplicate memory accesses to the internal buffers [DRO11]. Correspondingly, the MF can compute $1, 2$ or 3 outputs simultaneously as required and the actual number of data slots processed by the MF each time it is active is shown in <5>; and <6> indicates the data output from the MF. Every time the MF completes an output calculation, the corresponding value from the *intermediate buffer* is accessed to perform SIC_2, emptying it as shown in <7>. However, in this scenario, when using a reduced buffer of 1008 bytes the FTN mapper overwrites the previously written values before it has been used by the MF. This happens during the processing of data slot 12 by the FTN mapper and is highlighted in grey on <2> (WR to *intermediate buffer*) and <7> (RD from *intermediate buffer*). During this time, new values are written to addresses starting from 504 before the previous results are used up for SIC_2. Thus, all results calculated starting from this address will be incorrect resulting in wrong estimates of interference and in turn incorrect decoded bits. From Figure 5.11 it is seen that for $T_\Delta = 0.4$, the MF has to wait for 8 data slots for its inputs to be ready. In order to avoid new values being overwritten during SIC_1, it has to be prolonged in some way until MF has used the previous results. The proposed approach is to extend the memory size by appending a small buffer of 128 bytes, resulting in the total buffer size of 1134 bytes. By doing so, the WR accesses by the FTN mapper on the conflicting address is postponed to the next data slot by when the MF completes using the previous results without any data corruptions. This is shown by the conflict free address calculations between <3> (WR to *intermediate buffer*) and <7> (RD from *intermediate buffer*). Since the minimum memory size for $T_\Delta = 0.4$ accounts for the worst case scenario, all other configurations within the FTN system ($T_\Delta \geq 0.5$) can operate safely under this specification.

5.7.3 Interference canceled symbol buffer optimization by fixing the values of noise variance

The LLR calculation within the inner decoder is implemented as a multiplication between scaled interference canceled symbols ($\check{x}_{k,\ell} << 1$) and the inverse of the estimated variance (σ^2) i.e.,

$$LLR(\check{x}_{k,\ell}) = \frac{1}{\sigma^2}(\check{x}_{k,\ell} << 1).$$

The σ^2 used for LLR calculations is the same for all interference canceled symbols during a particular iteration. In the initial implementation, while the σ^2 was calculated the interference canceled symbols were buffered. Alternatively, the buffering and variance calculation can be eliminated altogether by using pre-defined values. This is already shown in [WHW00] for turbo decoders using max-log-MAP implementations. Here we apply this concept to FTN systems and evaluate the decoder performance. This improves the decoding speed as well as silicon area by reducing a large memory required for buffering. Further, in the case of FTN system the σ^2 is not just the variance of the noise but the effective variance of the noise as well as the interference arising due to FTN signaling [DRO11]. Here it is shown that, using fixed values of σ^2 is also applicable to FTN based systems and follows a similar trend as that in [WHW00].

Noise profile

Generally during FTN decoding, σ^2 is high during the initial iterations due to the interference introduced with FTN signaling as well as noise from the channel. As the iterations progress, the noise and interference is canceled out resulting in a cleaner signal and hence σ^2 becomes lower. This fact can be exploited to determine how values for σ^2 are set over the iterations. These different values of σ^2 can be set over one or several iterations and is referred to as 'noise profile'. Different noise profiles can be defined for different FTN configurations. In this work, simulations were performed by using a single noise profile for all FTN configurations. In this profile, a fixed value of σ^2 is used over a range of iterations, within the entire decoding cycle.

The chosen noise profile is $\sigma^2 = \{8, 4, 1\}$, where σ^2 is set to 8 in the first iteration, to 4 between iterations $2 - 4$ and from iterations $5 - 8$, σ^2 is set to 1. Figure 5.12 shows the bit error rate (BER) performance of different FTN configurations when using the above mentioned noise profile. The solid lines correspond to the BER performance of the FTN system when σ^2 is calculated in every iteration [DRO10], while the dash-dot lines are the BER performances

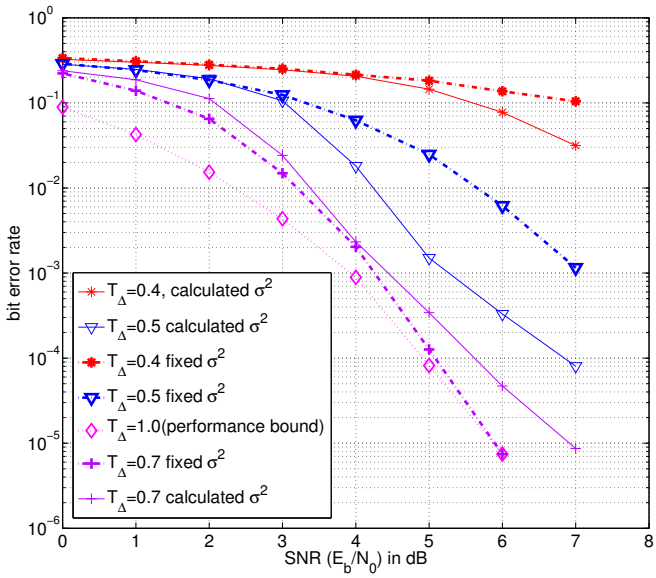

Figure 5.12: BER performance of the FTN decoder with fixed values of σ^2.

when using the pre-defined noise profile. The $T_\Delta = 1$ curve is the performance bound for the $(7, 5)$ convolutional code. It can be seen that for $T_\Delta = \{0.4, 0.5\}$, the performance degrades at higher SNRs. However, for $T_\Delta = 0.7$, it actually improves and runs very close to the performance bound. The reason for this behavior is the following. For $T_\Delta = \{0.4, 0.5\}$, the interference amongst the symbols are higher and the chosen variance is an under-estimate resulting in poorer performance. On the other hand for $T_\Delta = 0.7$, the interference due to FTN signaling is milder and the set variance being an over-estimation results in better decoding performance. The performance degradation can be overcome by having different profiles of noise densities for different T_Δ configurations. In order to find these profiles of noise densities a more elaborate study has to be performed. However, the conclusion is that by using pre-defined noise profiles in decoding of FTN modulated signals the 2kB *interference canceled symbols buffer* as well as the noise variance calculation block can be completely eliminated resulting in significant savings in silicon area.

Table 5.2: Resource utilization for the memory optimized FTN iterative decoder implemented in ST 65nm standard cell CMOS process.

Blocks	Logic area (μm^2)	Logic area (%age)	Memory (μm^2)	Memory (%age)
Inner Decoder	**108 952**	**62.7%**	**139 567**	**64.1%**
- Soft output calc	646	0.4%	-	-
- LLR calc	300	0.2%	-	-
- SIC/FTN Mapper	31 011	17.8%	65 739	30.2%
- SIC/Matched Filter	71 080	40.9%	25 387	11.7%
- SIC/Temp. buffers	5 910	3.4%	48 440	22.3%
Outer Decoder	**17 526**	**10.1%**	**36 976**	**17.0%**
Π **and** Π^{-1}	**44 068**	**25.3%**	**41 032**	**18.4%**
Global FSM	**3 086**	**1.8%**	-	-
FTN Decoder	***173 782***	100.0%	***217 576***	100.0%

5.8 Post-optimization results

The resource usage of the optimized FTN decoder is summarized in Table 5.2. Post optimization, the buffers within the SIC is reduced. However, the inner decoder still accounts for more than 60% of the overall area largely due to the use of several memories as well as multiple instantiations of the matched filter. However, there has still been a significant reduction in power and area primarily due to the memory optimization of the original design. The reduction in power and memory with supporting numbers are discussed in the following.

Power consumption

The memory optimized design is again evaluated for power consumption using the synthesized gate level netlist. Both designs were synthesized under the same constraints and power consumption is estimated using Synopsys™ PrimeTime™ while running the design at 200MHz. The estimate of the total power (dynamic and leakage) reduced from 44.7 mW to 25.1mW. A majority of the reduction is due to the memory optimization. Table 5.3 lists the breakdown in power consumption with respect to the logic and memory between the baseline and memory optimized implementations. The memory optimization included the reduction of large buffers within the SIC and optimize the usage of dual port RAMs into single port counterparts where ever applicable. It can

Table 5.3: Comparison of power consumption between baseline and Memory optimized design.

	Baseline [DRO11]	Memory optimized	Savings
Total Power	44.7 mW	25.1 mW	43.8%
- Logic	8.0 mW	6.5 mW	18.7%
- Memory	36.7 mW	18.6 mW	49.3%

Table 5.4: Area comparison between baseline and memory optimized implementations.

	Baseline [DRO11]	Memory optimized	Savings
Chip area	0.519 mm^2	0.391 mm^2	28.7%

be seen from Table 5.3 that memories accounted for 80% of the overall power and hence its optimization resulted in power savings of 43.8% when compared to the baseline implementation.

Memory requirement and chip area

The overall chip area and memory requirements between the baseline and memory-optimized architectures is summarized in Table 5.4 and Table 5.5. The *reconstructed symbol buffer* now uses a single port RAM instead of a dual port RAM as the requirement for simultaneous read/write access has been removed, providing 2.5 times reduction in memory area. The size of the *intermediate buffer* has been reduced by 45%. The *interference canceled symbol buffer* has been completely eliminated by using a fixed noise profile. The overall memory reduction achieved in the optimization of the inner decoder is 37% (4.86kB as against previously used 7.76kB). These optimizations have reduced the overall memory requirement of the inner decoder to the same order as that of a simple max-log-MAP decoder implementation for the $(7, 5)$ convolutional code (see outer decoder memory requirement in Table 5.5). However, no optimization on the outer decoder has been carried out.

The memory required in the inner decoder has decreased from 7.76 kB to 4.86 kB, nearly a 40% reduction. Consequently, the silicon core area came down from 0.519mm^2 to 0.391mm^2. With the improved memory architecture, the chip area is reduced by 28.7% of what was previously reported in [DRO11].

Table 5.5: Memory requirement for baseline and memory optimized architecture implemented in 65nm CMOS [ST].

Blocks	Baseline		Memory optimized	
	Memory size	Memory area (μm^2)	Memory size	Memory area (μm^2)
Inner decoder	**7.76 kB**	**233 979**	**4.86 kB**	**147 103**
- FTN mapper	1.01 kB	65 739	1.01 kB	65 739
- matched filter	0.75 kB	36 597	0.75 kB	36 597
- Interm. buf	2.00 kB	43 881	1.10 kB	27 203
- Int. canc. buf	2.00 kB	43 881	0.0 kB	0
- Reconst. buf	2.00 kB	43 881	2.00 kB	17 564
Outer decoder	**4.92 kB**	**46 260**	**4.92 kB**	**46 260**

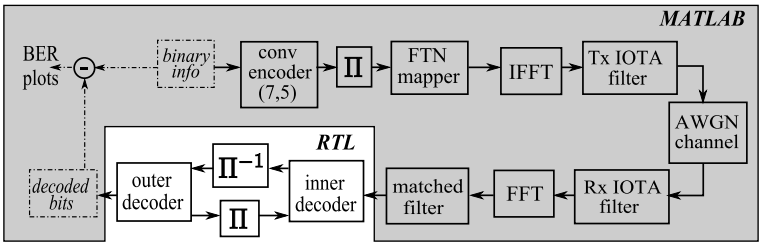

Figure 5.13: System setup for simulation of the RTL implementation of FTN iterative decoder using the MATLAB transceiver model.

5.9 RTL verification using MATLAB system model

The BER performance of the RTL implementation of the FTN decoder is evaluated to compare the design with the reference MATLAB™ model. Figure 5.13 shows the system setup for the simulation of the RTL implementation through MATLAB. The RTL is simulated in ModelSim™ by providing stimulus from MATLAB that corresponded to different T_Δ. 200 random binary information blocks were decoded for each T_Δ and at each SNR. Figure 5.14 shows the average BER performance of the implemented FTN decoder with respect to the performance of floating and fixed point simulations from MATLAB. The performance from the RTL implementation is consistent with the fixed and floating

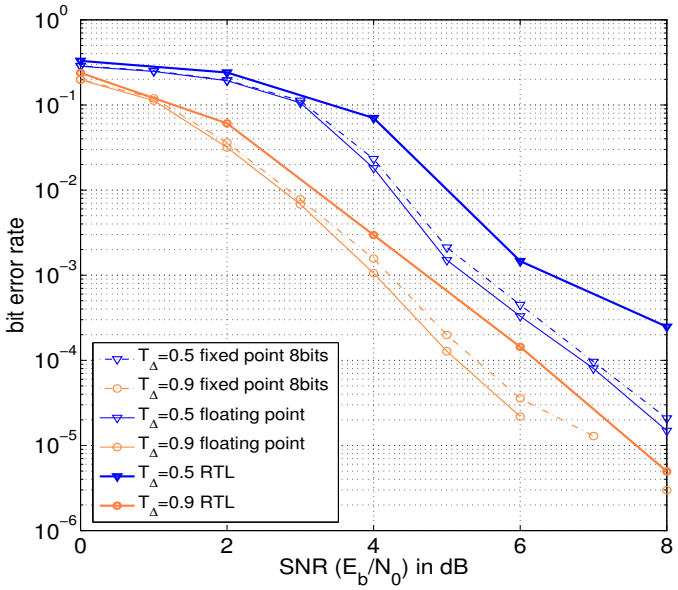

Figure 5.14: BER performance from RTL simulations showing conformance with floating and fixed point MATLAB reference models.

point models but with some degradation in performance. This is due to the pseudo-random interleaver [3GP07] used in the RTL implementation compared to the random interleaving in MATLAB. The other reason is that the number of realizations over which the decoding is averaged is limited for RTL due to simulation runtime. The performance is shown to indicate functional correctness of the RTL implementation of the FTN decoder.

5.10 FTN signaling in transceivers

Generally transmitters and receivers are realized in hardware as transceivers. This is because either transmit or receive signals are processed at any given point of time. This also gives the advantage of resource sharing between transmitter and receiver. A classic example of this is the multicarrier modulation/demodulation realized using IFFT/FFT in an OFDM (multicarrier) system. A single entity can be used to perform both IFFT/FFT. This also provides an advantage of lesser silicon usage and less hardware overhead. Similarly, while

designing FTN signaling specific blocks it has been taken into account how these blocks can be time shared between transmitter and receiver as well as performing different functionalities within the receiver.

Figure 5.15 shows the block diagram of a transmitter and receiver implemented as separate chains within a device. The FTN processing blocks are intentionally designed so that they can be reused. However, as can be seen from the figure, some blocks specific to FTN signaling are duplicated when realized as separate entities. The intention of designing the processing blocks in both transmitter and receiver is that, when they are implemented as a single entity many of the blocks can be time shared. Such a block diagram showing the time shared resources between the transmitter and receiver chain is shown in Figure 5.16. When the transmitter is active, the FTN mapper is enabled together with the rest of the transmit processing blocks. During decoding of FTN signals in the receiver, either the MF alone or both FTN mapper as well as MF are active. This depends on whether reconstruction of FTN symbols is being carried out or if it is the inner decoder with SIC that is processing the data. The transmit and receive IOTA filter can also be designed to be a single processing block that can perform both functions. In summary, and as evident from Figure 5.16, the overhead from the FTN specific processing blocks is the requirement of one instance each of FTN mapper, MF and IOTA filter for the entire FTN transceiver chain.

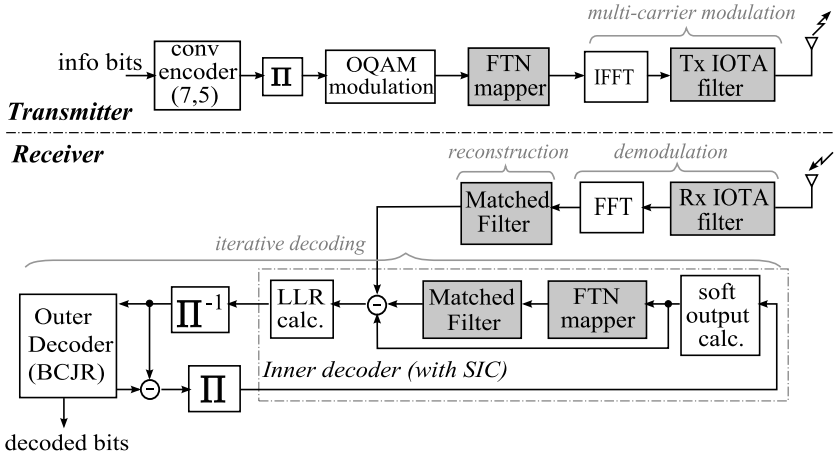

Figure 5.15: Block diagram of the FTN transmitter and receiver as realized separately.

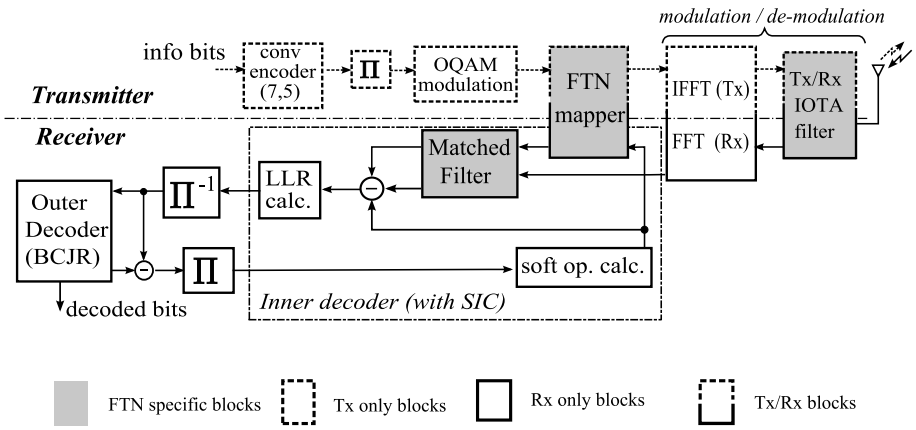

Figure 5.16: Block diagram of multicarrier FTN transceiver showing reuse of FTN specific processing blocks.

5.11 Summary

This chapter has dealt with the hardware architecture of the decoder responsible for decoding of FTN modulated signals. First, the matched filter for FTN symbol reconstruction is presented. The reconstructed symbols are then iteratively decoded by passing the LLRs of the received symbols between the inner and the outer decoders. The inner decoder is a symbol-by-symbol MAP decoder with successive interference cancellation. The SIC is realized as a cascade of mapper and MF, which are used to realize other functionality. The SIC being a significant part of the inner decoder reuse of processing blocks helps in keeping hardware overhead low. This iterative decoder is implemented for both FPGA and in a 65nm CMOS ASIC process. The logic and memory requirements of the inner decoder is discussed and analyzed in relation to the outer decoder which is a conventional implementation, the max-log-MAP decoder. The operating speed, achievable throughput and estimated power consumption of the ASIC implementation is also presented.

In order to reduce both power and silicon area for the intended implementation of the decoder, architectural optimizations are carried out for memory reduction. The complete implementation of the post-optimized architecture was verified in RTL and the design was taped out. Finally, all blocks specific to FTN signaling is analyzed in the perspective of a transceiver and how several blocks can be time multiplexed is depicted.

Chapter 6

FTN decoder chip: Measurements and results

This chapter presents the details of the implemented FTN decoder chip. There has been several challenging aspects that had to be overcome during the ASIC implementation of the FTN decoder. These have come up at various stages of the implementation process, from requirement to optimize for area at the architectural level to the number of input/outputs (IOs) at the physical implementation stage. These aspects are briefly discussed in order to provide the reader an overview of the not very obvious challenges faced during the ASIC implementation.

The silicon die area available for fabrication from ST microelectronics [ST] for research projects at the Dept. of EIT is $1\,\mathrm{mm}^2$. Within this area several designs are planned and fit so as to make the best use of the silicon area. Though the area reported for the FTN decoder in Section 5.8, was about $0.4\,\mathrm{mm}^2$, it only constituted the logic and memories. Apart from this, a physical implementation of an ASIC requires a bigger area so that, along with the logic and memories, IO pads, power lines, etc., can be placed. It also requires that there is some margin in the core area of the chip to include buffers and resize the logic cells so that timing can be met and the signals can be routed. With all these considerations, the resulting area came up to $0.8\,\mathrm{mm}^2$.

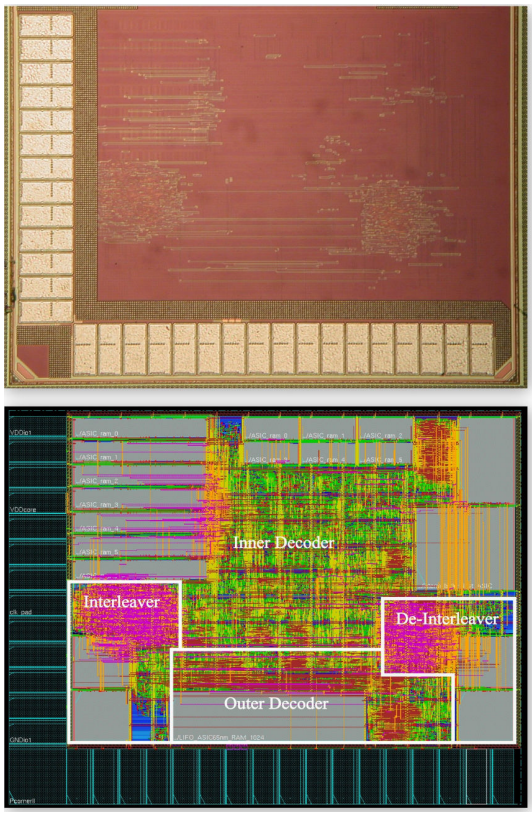

Figure 6.1: Chip photo (top) and the layout (bottom) showing the functional blocks of the FTN decoder implemented in 65nm CMOS.

A further constraint with the IOs is that the pad area should be 20% of the overall die area in the ST 65nm CMOS process. This translated to 30 IO pads that could be used within an area of 0.8 mm^2 restricting free usage of IOs for functional as well as debug lines. In order to maximize the core area as well as number of pads, the IO planning is intentionally placed only to the left and bottom of the design as shown in Figure 6.1. Though this approach is unconventional, it does not create functionality or fabrication issues. The implemented design measures 1 mm in width and 0.8 mm in height. The key functional blocks in the iterative decoder are also highlighted in the layout.

Table 6.1: Time-Frequency spacing configuration for FTN signaling.

TF_spacing[2:0]	$T_\Delta F_\Delta$
0 0 0	0.4
0 0 1	0.5
0 1 0	0.6
0 1 1	0.7
1 0 0	0.9
1 0 1	1.0

6.1 Chip configurations

The FTN decoder chip has been designed to run under several different configurations to improve or evaluate the performance of the decoder under these configurations. The configurations are 'TF spacing', 'Noise profile', and 'number of iterations'. TF spacing is used to configure the FTN decoder to decode the received FTN modulated block that has been transmitted with a certain $T_\Delta F_\Delta$.

6.1.1 Time-frequency spacing configuration

The TF spacing is configured using 3 bit inputs to the FTN decoder chip and Table 6.1 lists the bits that are to be specified for the decoder to operate under a certain $T_\Delta F_\Delta$. The range of time-frequency spacings ($T_\Delta F_\Delta$) that are configurable for the FTN decoder chip are $\{0.4, 0.5, 0.6, 0.7, 0.9, 1.0\}$.

6.1.2 Noise profile configuration

The noise profile configuration is introduced to eliminate the noise variance calculation unit as well as the large memory associated with it. The 'noise profile' is used to specify a certain profile that defines the trend of the variance of the noise plus the interference over the iterations. In Section 5.7, it was shown that by fixing the values of noise variance, memory and in turn power can be reduced significantly. However, the decoder performance while using different $T_\Delta F_\Delta$ may vary with the choice of 'noise profile'. In the implemented FTN decoder, 4 different noise profiles can be chosen for decoding of the received symbols.

Table 6.2 shows different noise profiles that can be evaluated by setting the noise_profile[1:0] bits. The value of noise variance used over the iterations is

Table 6.2: Noise profile configurations.

noise_profile[1:0]	variance (σ^2) over {1,2:4,≥5} iterations
0 0	{1, 1, 1}
0 1	{8, 4, 1}
1 0	{4, 2, 1}
1 1	{16, 4, 1}

shown in the second column of the table. In each noise profile configuration, the 1st value indicate the variance during the first iteration, the 2nd during iterations $2-4$, and the 3rd for iterations 5 and beyond. Any configuration of noise profile can be used to decode the FTN modulated signals transmitted using any of the TF spacing configurations[1]. Further, the values of the variance (σ^2) are chosen as powers of 2 in order to avoid the division operations while calculating LLRs. By choosing σ^2 as powers of 2, the division operation turn out to be simple shifts and come with no extra hardware cost.

6.1.3 Configuring number of iterations for decoding

The decoder is capable of processing the received information block over a range of iterations (from 1 to 16). In Chapter 2, it was shown that, 8 decoding iterations proved to be good enough to achieve a reasonable BER performance. However, later in Chapter 3 the decoding was performed by adapting the number of iterations to the SNR. This aspect is taken into account while designing the FTN decoder and included as a feature in the implementation. Table 6.3 shows the range of iterations supported by the FTN decoder chip and the corresponding bit configuration that is to be set to from outside in order operate the decoder using a certain number of iterations.

[1]However, the decoder generally performed better with noise profile $\sigma^2 = \{8, 4, 1\}$ during the evaluation presented in Section 5.7

Table 6.3: Configuring number of iterations.

no_of_iterations[2:0]	Number of iterations
0 0 0	1
0 0 1	2
0 1 0	4
0 1 1	6
1 0 0	8
1 0 1	10
1 1 0	12
1 1 1	16

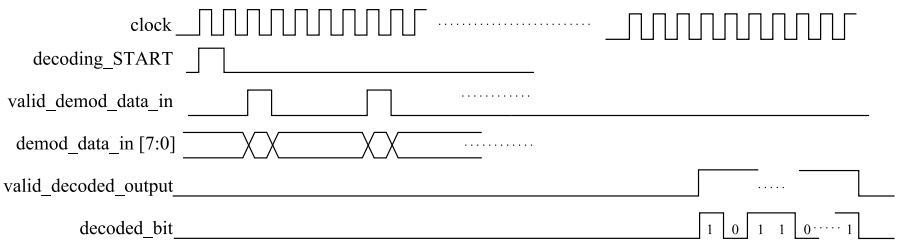

Figure 6.2: Illustration of the sequence of steps during the decoding of a received information block.

6.1.4 Sequence of steps during the operation of the FTN decoder

Th following steps describe the sequence of operations that are carried out during the decoding of a block of received information with the help of the illustration in Figure 6.2.

1. The decoder is configured to operate under the specified *no_of_iterations*, *TF_spacing* and *noise_profile*.
2. The control signal *decoding_START* initiates the start of iterative decoding by indicating the decoder about the arrival of data.
3. A *valid_demod_data_in* signal indicates when a valid demodulated data sample *demod_data_in* is present on the input.
4. After having received the last data sample, the decoder runs through the

specified number of iterations decoding the data that is transmitted using a given $T_\Delta F_\Delta$ and using the specified noise profile.

5. When the decoding is complete, the *decoded_bit* output is indicated by a 1 on the *valid_decoded_output* line.

6. LLRs, one bit each from the inner and the outer decoder are tapped out for identifying any discrepancies in the component decoders during the decoding process. The *inner_decoder_LLR* refers to the LLR from the inner decoder while, *outer_decoder_LLR1* refers to the first of the two LLRs from the outer decoder. Though the LLRs are soft information and are several bits wide (8 bits), only the MSB is tapped out as it is enough to know the sign of the LLR to identify the bit being decoded. It is also possible to write out all the bits of the LLR in a serial fashion. However, this requires a faster clock to serialize the LLRs introducing extra effort in designing the chip with multiple clock domains. Even if the LLRs are read out serially, it can only be analyzed, and any problems that may arise still cannot be fixed in the ASIC. Furthermore, due to the constraint on the number of pads allowed in the available die area, the debug pins were chosen to be the sign bits of the LLRs.

Figure 6.3 shows snapshots from 3 separate measurements using the Logic Analyzer during the decoding of a block of received information with 1, 6 and 10 iterations. The lower most signal in each snapshot is the *decoding_START* pulse triggering the beginning of the decoding cycle. The *inner_dec_LLR* and *outer_dec_LLR1* are LLRs from the inner and the outer decoders that are passed back and forth between the decoders until the configured number of iterations is reached. After completion of the decoding cycle, the *valid_decoded_bit* line is set to 1 indicating the availability of the decoded bit on the *decoded_bit* output.

In the topmost waveform indicating decoding with 1 iteration, soon after the *decoding_START* is triggered, the inner decoder starts calculating the LLRs. At the end of the inner decoder computations, the outer decoder begins its calculations and writes out the decoded bits and indicating valid outputs by asserting the *valid_decoded_output* line. In the lower two sets of waveforms, the process of inner and outer decoding cycles run for 6 and 10 iterations respectively. In the last iteration, the outer decoder performs hard decision and outputs the decoded bits.

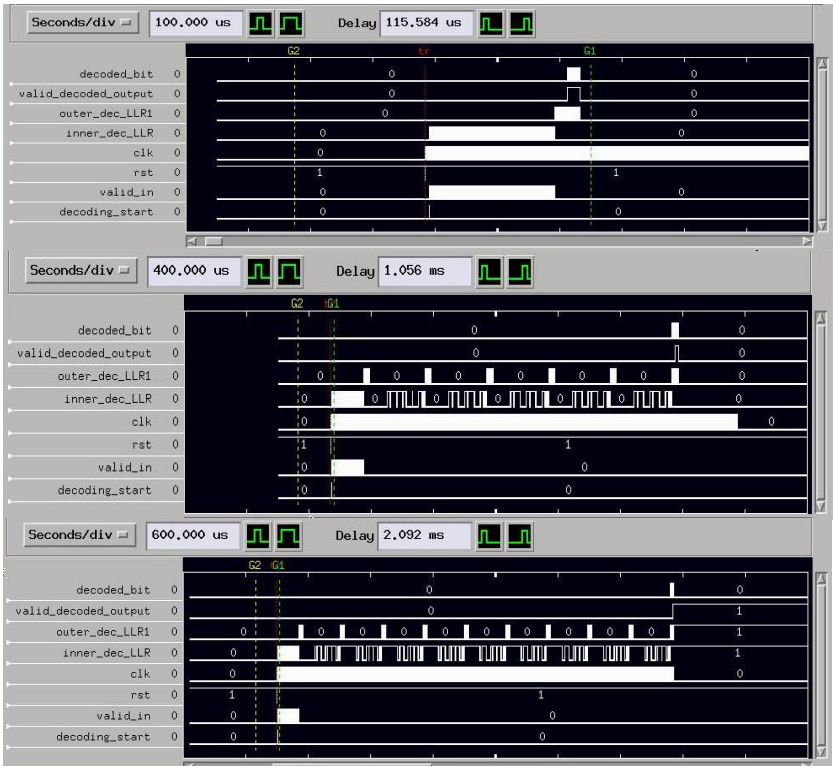

Figure 6.3: Snapshot from the Logic Analyzer for 3 separate measurements showing 1, 6 and 10 iterations.

6.2 Measurement results

One of the main goals of this work is to prove the feasibility and implementability of FTN signaling in hardware. The work so far has encompassed designing a system model of the FTN transceiver, evaluating its performance, converting the system model to operate on fixed point numbers as in hardware, make design choices on the parameters in the system and implementation of the FTN decoder in ASIC. The final step is to measure the silicon implementation of FTN decoder for its functionality and decoding performance in comparison with the MATLAB reference model.

The FTN decoder is setup for measurement using the system model shown in Figure 6.4. The transmitter, channel and parts of the receiver is retained from MATLAB system model while the FTN decoder is interfaced to the rest

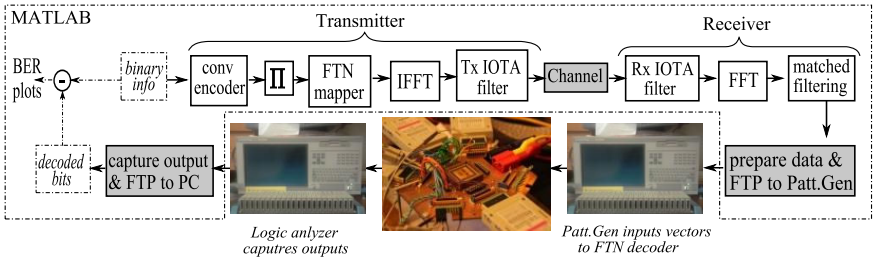

Figure 6.4: Setup for the measurement of the FTN decoder chip.

of the system through the pattern generator and logic analyzer. The MATLAB part consists of generating random binary information, passing it through the transmitter chain, the channel and some receiver blocks up to the MF. The SNR of the AWGN channel, number of decoding iterations, TF spacing and noise profile is set in the MATLAB model. The output from MF is provided to the FTN decoder chip by preparing the configuration and data specific to the pattern generator. The decoded data bits from the chip is captured by the logic analyzer which is read back into MATLAB and the decoded bits are extracted. The decoded bits are compared with the original random binary information generated for transmission. For each SNR and and TF spacing $T_\Delta F_\Delta$, Monte-Carlo simulation are performed to obtain the decoding performance in terms of average bit error rate.

Figure 6.5 shows the BER performance of the FTN decoder chip for $T_\Delta = \{0.6, 1.0\}$. The performance is shown in comparison with the MATLAB reference model using floating point number representation. The thick solid lines refer to the measured BER from the chip while the dash-dotted lines refer to that from the reference model. It should be noted that the MATLAB reference model uses the variance calculation while the FTN decoder chip incorporates variance calculation by using noise profiles. The noise profile used during the measurement is $\sigma^2 = \{8, 4, 1\}$. Due to the use of fixed noise profiles and the fixed point representation, the decoder performance deviates from the MATLAB reference by about 0.5 dB.

The comparison is brought out between the MATLAB floating point performance of the decoder and that measured in silicon to visualize the overall performance tradeoff when realizing FTN from algorithm to hardware. The major result from this work is that the FTN decoder has been tested and evaluated both at the system level as well as in a ASIC hardware. The implemented device is verified in silicon and is shown that it can to operate in both FTN ($T_\Delta = 0.6$) and orthogonal ($T_\Delta = 1.0$) modes.

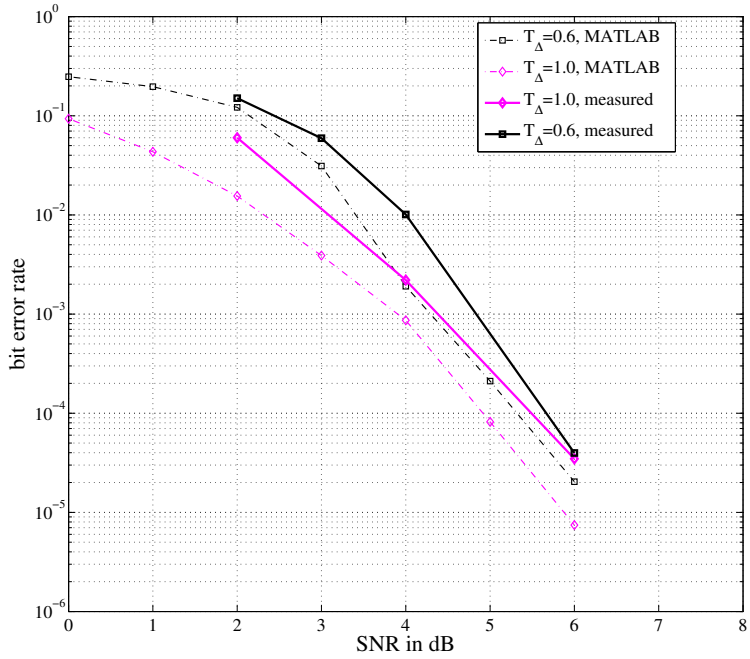

Figure 6.5: Measured performance of the FTN decoder vs. the BER performance from MATLAB simulations for $T_\Delta = \{0.6, 1.0\}$.

Speed and throughput

Due to a hard constraint on the total available die area, the placement and the routing density turned out to be much higher resulting in congestion that in turn reduced maximum achievable clock frequency. Compared to the reported clock period of 3ns at the netlist stage, the post place and route netlist could achieve a clock period of 9.25ns that corresponded to a safe operating clock frequency of 100MHz. It is shown in [DRO11], that the FTN decoder could run at a clock frequency of 333MHz and achieve a raw data throughput of 6.4Mbps or 3.2Mbps of information throughput (due to the use of rate 1/2 outer code). With a factor of 3.33 reduction in speed, the FTN decoder running at 100MHz is able to achieve close to 1Mbps (0.969Mbps) of information throughput. However, we believe that by relaxing the die area constraint, the congestion can be reduced resulting in better timing and an operational clock frequency of 333MHz can be achieved.

6.3 Summary

Measurements from the silicon implementation of the FTN decoder chip is presented in this chapter. The chip is implemented using ST 65nm standard cell CMOS process and occupies a die area of 0.8 mm^2. It has been designed to operate under various configurations including the number of iterations, different noise profiles and different time-frequency spacings ($T_\Delta F_\Delta$). It has been demonstrated with BER performance curves that the actual measured performance deviates by about 0.5 dB when compared to the MATLAB reference model. Further, this deviation includes the tradeoffs from moving over to fixed point representation as well as using fixed set of noise profiles. Finally, it has been demonstrated that both FTN and orthogonal signaling is achievable using the same decoder and more importantly the feasibility of realizing FTN signaling in hardware.

Chapter 7

IOTA pulse shaping filter in FTN multi-carrier systems

7.1 Introduction

IOTA stands for Isotropic Orthogonal Transform Algorithm and is derived by orthogonalizing a Gaussian pulse in both time and frequency [Ala01, LFAB95]. Such pulses are generally called root-Nyquist self-transform pulses; *root-Nyquist* because the square of the pulse results in a Nyquist pulse [Nyq28, PS08] and *self-transform* because the pulse and its Fourier transform are identical. The IOTA pulse, shown in Figure 7.1, derives its name due to its isotropic nature in both time and frequency domains.

Conventionally, an OFDM system's robustness in multipath channels is due to an extended length of the OFDM symbol by zero padding (ZP) or by introducing a cyclic prefix (CP), hereafter referred to as ZP-OFDM and CP-OFDM respectively. The ZP/CP is designed to be equal to or longer than the impulse response of the channel in order to combat the multipath effects. Typically it accounts for 10−25% of the OFDM symbol [BBK04, 3GP10]. This part of the OFDM symbol contains no or redundant information and results in reduced bandwidth efficiency. An alternative approach is through the use of pulse shaping filters that employ well localized basis pulses but comes at a cost of higher processing complexity. These systems are also commonly referred to as filter bank based multicarrier modulation systems and use offset-QAM as

[0]The hardware architecture and its implementation of IOTA pulse shaping filter presented in this chapter has been carried out in co-operation with Shahid Mehmood, former MS student at the Dept. of EIT.

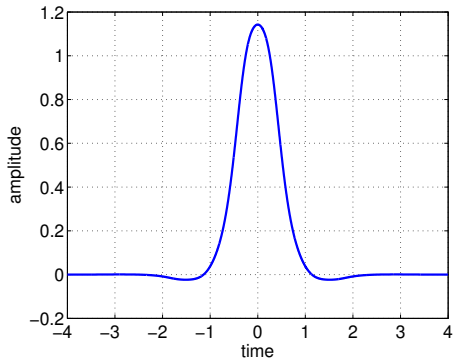

Figure 7.1: IOTA pulse.

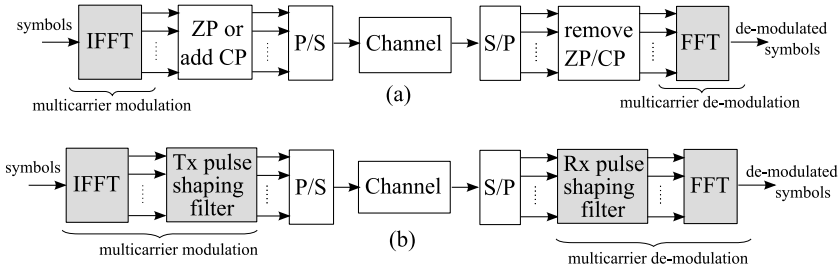

Figure 7.2: Comparison between ZP/CP OFDM and IOTA pulse shaped OFDM.

the preferred type of modulation [Hir81, Cha66, BD74, BDH99].

Figure 7.2 shows a comparison between ZP/CP and IOTA based OFDM system. While the ZP/CP based system simply appends zeros or a cyclic prefix, an IOTA based system performs transmit and receive filtering using the IOTA pulse. The pulse shaping filter together with the IFFT/FFT forms the multicarrier modulation/de-modulation block.

The IOTA pulse has previously appeared in both literature and as patents. In [LFAB95, LGA01] the IOTA filter is described and how it can be used to avoid the cyclic prefix in fading channels. The patent in [CLJ06] is exclusive to the IOTA pulse, while a more general patent of root-Nyquist self-transform pulse shaping filters can be found in [Den09]. The IOTA has also found its way into the 3GPP standard [3GP04, JL03] as it has been shown that it can avoid the cyclic prefix altogether. Hardware architectures of IOTA based pulse shaping filters can be found in e.g. [SZCP07, SCZP07].

7.1.1 IOTA in FTN systems

IOTA pulse shaping filters were used to reduce hardware overhead and increase the performance in multicarrier systems based on faster-than-Nyquist (FTN) signaling [DRAO09]. Being a part of the FTN system and contributing to complexity reduction, it is important to have an hardware efficient architecture for the filter itself so as to keep the overhead moderate. The primary intention of using an IOTA basis in our FTN system is due to the time-frequency localization property resulting in fewer projection coefficients as compared to the rectangular basis [DRAO09].

In this work, we have obtained the discrete time prototype of the IOTA filter by first constructing the continuous time pulse and then truncating and discretizing it. It is well known that a pulse, orthogonal in continuous time, need not satisfy orthogonality after truncation and discretization in time [SSL02, SSP06]. Hence, the filter length has been kept long enough such that, though orthogonality is lost, the resulting interference (ISI and ICI) is small and hence ignored. One way of directly obtaining the discrete prototype filter is proposed in [SSL02], while an alternative approach using discrete Zak transform can be found in [BDH03]. The approach in [SSL02] requires optimization of several parameters and is beyond the scope of the current work. Obtaining the prototype filter through truncation and discretization is found to be satisfactory for the system under consideration.

7.2 Functional description of transmit/receive IOTA filter

The transmit and receive functionality of the IOTA filter is shown conceptually in Figure 7.3 (a) and (b) respectively. At the transmit side, the OFDM symbol after the multicarrier modulation (IFFT) is replicated by an oversampling factor, denoted by $2L$. This factor is equal to the ratio between the length of the filter and the number of sub-carriers (N) and generally $N >> 2L$. Oversampling at the output of IFFT is required as the IOTA pulse has a longer time support than the OFDM symbol period T [LFAB95]. In each time instance, the replicated OFDM symbol is weighted by the filter coefficients and accumulated with a part of the previously weighted and summed OFDM symbols. This is demonstrated in Figure 7.3(a) for 3 OFDM symbols.

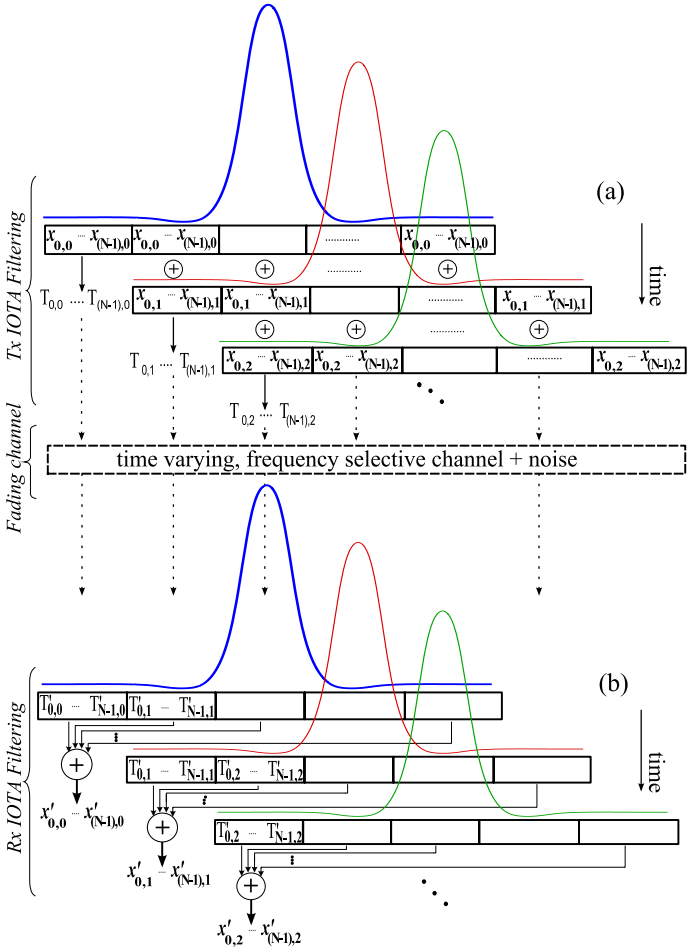

Figure 7.3: Illustration showing the functionality of IOTA transmit and receive filter.

This transmission of IOTA pulse shaped signals shown in Figure 7.3(a) can be mathematically described as

$$T_{k,m} = \sum_{\ell=0}^{2L-1} x_{k,\ell-m} \cdot I_{k+\ell N}, \tag{7.1}$$

where x correspond to the input samples that are multiplied by IOTA filter coefficients I, $k = \{0, 1, \ldots (N-1)\}$ is the index of the sub-carriers and $m =$

$\{0, 1, \dots\}$ is the time index. The resulting output samples are denoted by T. The filter outputs are then transmitted over a frequency selective wireless channel. At the receiver, symbols are weighted by the same filter coefficients and the N samples corresponding to one OFDM symbol are accumulated from several received symbols as demonstrated in Figure 7.3(b) for the same 3 OFDM symbols. The receiver functionality shown in Figure 7.3(b) can be formulated as

$$x'_{k,m} = \sum_{\ell=0}^{2L-1} T'_{k,\ell+m} \cdot I_{k+\ell N}, \tag{7.2}$$

where T' correspond to the received samples while I, k and m have the same meaning as previously described for Eqn. (7.1).

7.3 Hardware architecture

This section describes two hardware implementation approaches with respect to requirements for throughput and hardware area. Their pros and cons with respect to the IOTA filter and the entire FTN system are investigated. This is followed by a proposed time multiplexed architecture that can be used for pulse shaping in both transmitter and receiver.

7.3.1 Hardware mapped architecture

An effective approach for implementing a fully parallel IOTA filter is by decomposing it into filterbanks [Fli94, SSL02]. To be able to supply the filter with sufficient data, a parallel implementation of the IFFT providing N outputs simultaneously is assumed. The architecture of the transmit filter shown in Figure 7.4 refers to a $2NL$ tap filter divided into N filterbanks consisting of $2L$ taps each. A column in the figure refers to one such filterbank and the filter coefficients $I_0 - I_{2NL-1}$ are distributed across the filterbanks. The first bank consists of coefficients $I_0, I_N, I_{2N} \dots I_{2NL-N}$, the second with coefficients $I_1, I_{N+1}, I_{2N+1} \dots I_{2NL-N+1}$ and so on. For the filter length under consideration, a hardware mapped implementation according to Figure 7.4 would require 1024 multipliers, adders and registers. It is well known that fully parallel implementations become prohibitive for large filters, and that the hardware mapped approach is only feasible for multicarrier systems with relatively few sub-carriers. However, here it is presented in order to compare it with the proposed architecture in terms of e.g. area and speed. Results on the resource usage for the parallel implementation is provided in later sections. At the receiver side, a parallel implementation of the IOTA filter can be realized by

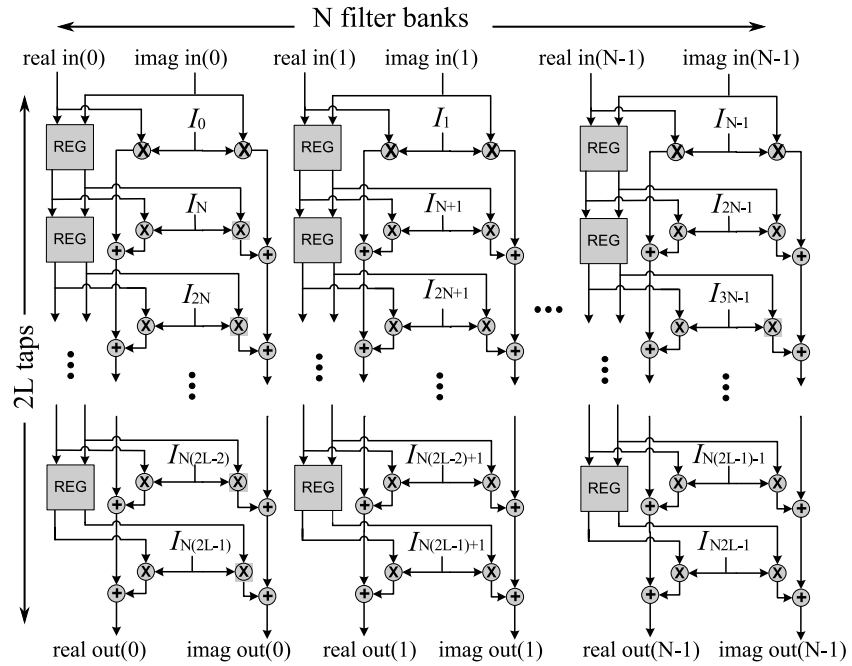

Figure 7.4: Parallel implementation of transmit IOTA filter (transposition results in receive IOTA filter).

simply transposing the transmit filter in Figure 7.4. Though the description and the architectures for the IOTA filter is mostly restricted to the transmitter in this paper, it equally holds for the receive filter.

Generally, the implementation of the IFFT block preceding the IOTA filter is not fully parallel, especially when the number of sub-carriers are large. This would make the filter structure of Figure 7.4, a parallel implementation, not well conditioned with the IFFT. Pipelined FFTs that provide 1 or a few outputs per clock cycle are more realistic from area and implementation point of view. In our study we have assumed that the IFFT/FFT implementation provides 1 output per clock cycle. However, the proposed filter architecture can be modified to support other input rates.

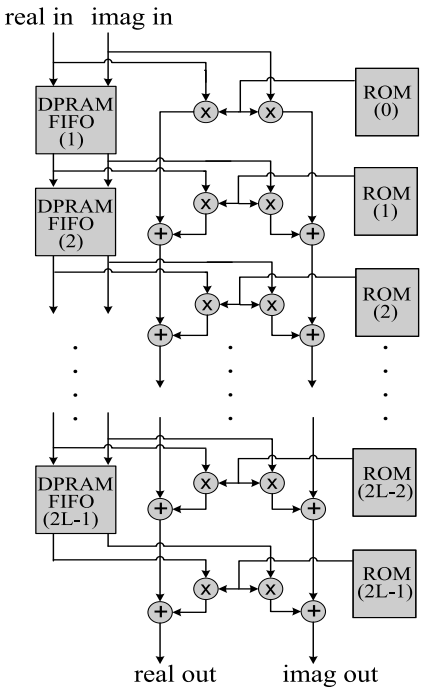

Figure 7.5: Horizontally folded architecture.

7.3.2 Time multiplexed architectures

When the length of the filter is large, as in our case, time multiplexed architectures are chosen over fully parallel implementations. Folding of the filter will reduce the arithmetic resources and FIFOs/RAMs can be used for storage, thus sacrificing throughput for area. The architecture presented in Figure 7.4 can be time multiplexed by folding it either horizontally or vertically.

Horizontal folding refers to time multiplexing one row of resources in Figure 7.4 using a complex multiplier, an adder and a FIFO. In this case the N parallel inputs are processed one at a time. This results in $2L-1$ FIFOs of size N and $2L$ complex multipliers and $2L-1$ adders. A basic hardware architecture of the transmit IOTA filter using a horizontally folded scheme is presented in Figure 7.5. However this architecture, similar to that in [SZCP07], require dual port RAMs (dp-RAMs) for FIFOs. dp-RAMs are required because with every incoming data, it has to be stored in the RAM while at the same time another value has to be read out in order to keep the arithmetic units fully utilized. dp-RAMs tend to occupy considerably more space than their single

Table 7.1: Arithmetic and storage complexity for folded and parallel architectures.

Resources	Parallel implementation (Figure 7.4)	Horizontally folded (Figure 7.5)	Vertically folded (Figure 7.6)
Multiplier: $2(w_i+w_c)$	$2NL$ (fixed)	$2L$	N
Adder: $2(w_i+w_c)+m$	$N(2L-1)$	$2L-1$	N
Registers: $2w_i$	$N(2L-1)$	-	-
No. of RAMs	-	$2L-1$	N
- RAM size (bits)	-	$N \times 2w_i$	$2L \times 2w_i$
No. of ROMs	-	$2L$	N
- ROM size (bits)	-	$\left(\frac{N}{2}+1\right) \times w_c$	$(L+1) \times w_c$

port counterparts. In the proposed architecture, FIFOs are optimized to use single port RAMs (sp-RAMs) without sacrificing performance, and is described in later sections.

Vertical folding refers to the architecture shown in Figure 7.6, when one column of resources in Figure 7.4 is time multiplexed resulting in N FIFOs of size $2L$ and N complex multipliers and adders. In this architecture the N parallel inputs to the filter banks remain.

7.3.3 Complexity Analysis

The arithmetic and storage complexity for the folded and the parallel implementations presented in the previous section is summarized in Table 7.1. The operations are performed on complex data except that the ROMs store the coefficients of the IOTA pulse which are only real. w_i refers to the wordlength of the real/imaginary part of the complex input and w_c the coefficient wordlength. Hence, for full precision, the width of the complex multipliers and adders are $2(w_i + w_c)$ and $2(w_i + w_c) + m$ respectively, where $m = \lceil log_2(2L-1) \rceil$ is used as guard bits to avoid overflow. A fully parallel implementation requires $2NL$ fixed multipliers and registers along with $N(2L-1)$ adders. The horizontally folded architecture requires $2L$ variable multipliers, $(2L-1)$ adders, RAMs and $2L$ ROMs storing $\frac{N}{2}+1$ values in each. On the other hand, the vertically folded architecture requires N variable multipliers, $(N-1)$ adders, N RAMs and ROMs capable of storing $(L+1)$ values in each.

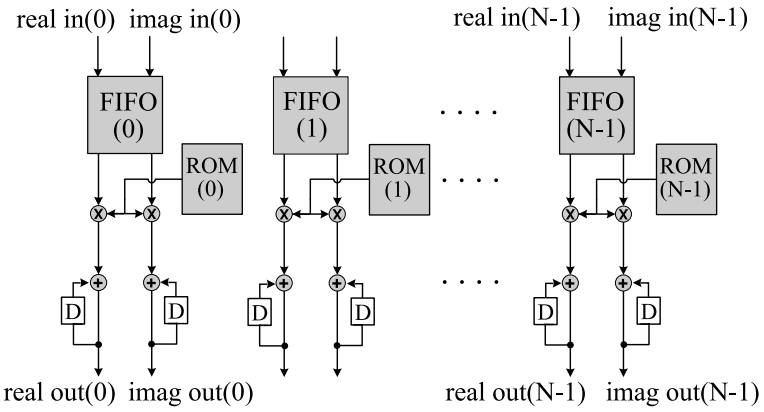

Figure 7.6: Vertically folded architecture.

Tables 7.2 and 7.3 exemplify the arithmetic and memory requirement for the horizontally and vertically folded architectures for two cases, $N = 128, 2L = 8$ and $N = 1024, 2L = 4$. This is in order to be able to better analyze the arithmetic and memory requirements. It is obvious from the two tables that the horizontally folded architecture is preferable when it comes to arithmetic complexity. This is especially evident for systems with large number of sub-carriers. It is also seen that the vertically folded architecture requires a large number of RAMs/ROMs of small size. For the example in Table 7.3 it is obvious that usage of RAM is not an attractive solution. Smaller size memories are more efficient when based on register than using RAM macros [MRB10]. However, realization using register banks brings back the storage count to that in a fully parallel implementation. Hence vertical folding does not result in area reduction by the folding factor in the filter.

7.3.4 Impact of the filter architecture choice on IFFT implementation

This section provides an overview of architectural issues arising while implementing the IFFT together with the IOTA filter. Most of these arguments hold good in the general sense. The actual implementation choice such as the choice of radix in IFFT, number of sub-carriers etc, also has an impact. The focus here is to motivate the architectural choice for the filter implementation and not the IFFT itself.

The following analysis on the time multiplexed IFFT implementation holds good only for the logic/arithmetic units in the IFFT, while it is not applicable

Table 7.2: Arithmetic and Memory requirement in the time-multiplexed architectures for $N=128, 2L=8$.

Resources	Horizontally folded (Figure 7.5)	Vertically folded (Figure 7.6)
Multipliers:	8	128
Adders:	7	128
No. of RAMs	7	128
Size of each RAM	128	8
No. of ROMs	8	128
Size of each ROM	65	5

Table 7.3: Arithmetic and Memory requirement in the time-multiplexed architectures for $N=1024$, $2L=4$.

Resources	Horizontally folded (Figure 7.5)	Vertically folded (Figure 7.6)
Multipliers:	4	1024
Adders:	3	1024
No. of RAMs	3	1024
Size of each RAM	1024	4
No. of ROMs	4	1024
Size of each ROM	513	3

to storage as it depends on several other factors. Both time-multiplexed filter architectures achieve the same functionality, but impose different constraints on the implementation of the preceding IFFT block. Since the horizontally folded architecture has a throughput of one sample per clock cycle, the IFFT realization can be time multiplexed by a factor of N. If A is the area of an N-point hardware mapped IFFT, then with horizontally folded filter implementation the IFFT area will be

$$H_{\text{IFFT area}} = \frac{A}{N}.$$

In the vertically folded case, the filter requires $\frac{N}{2L}$ samples per clock cycle (i.e., N inputs every $2L$ clock cycles). Hence, for the vertically folded filter the

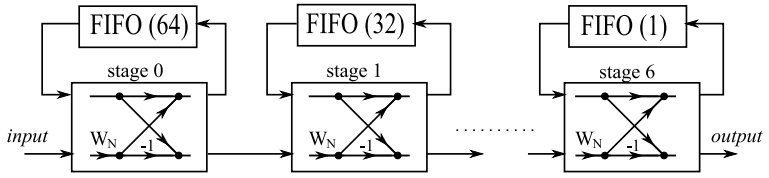

Figure 7.7: 128-point IFFT for horizontally folded IOTA filter.

IFFT can be time multiplexed by a folding factor of $2L$ and still match the throughput rates between the two. Thus, the area of the IFFT implementation for the vertically folded case will be

$$V_{\text{IFFT area}} = \frac{A}{2L}.$$

Comparing the two folded IFFTs with respect to area, we can say that

$$V_{\text{IFFT area}} = \frac{N}{2L} \cdot H_{\text{IFFT area}},$$

and since $N >> 2L$, the area savings in the H_{IFFT} will be much more than V_{IFFT}. In the strict sense, the folding factor for the IFFT in the vertically folded case will be the number of butterfly stages, i.e., $log_r(N)$ with r depending on the radix used in the implementation. Actual implementations involving both IFFT and the IOTA filter will have to consider the folding factor for the IFFT ($log_r(N)$ versus $2L$) so as to match the throughput rates with the IOTA filter.

The number of sub-carriers in the multicarrier system being $N = 128$, we look at the implementation of an 128-point IFFT. Radix-2 butterfly units are assumed in the implementation for the sake of simplicity in the analysis. Figure 7.7 refers to an IFFT implementation when horizontal folding is used in the IOTA filter. This is the well known pipeline architecture also referred to as Single-path Delay Feedback (SDF) architecture [WD84, HT98]. For $N = 128$ the implementation requires 7 ($log_2(N)$) butterfly stages and 127 $\left(\frac{N}{2^1} + \frac{N}{2^2} + \cdots \frac{N}{2^{log_2(N)}}\right)$ memory locations in total. When using the vertically folded IOTA filter, the corresponding IFFT is shown in Figure 7.8. The arithmetic resources required are much larger (64 ($\frac{N}{2}$) butterflies) and the memory accesses are also more complicated. With 64 butterfly units, all 128 outputs of a certain stage of IFFT calculation are available at the same time. However, saving all the results at once while using a RAM is not possible. Furthermore, the values stored in memory that needs to be provided to the butterfly units depend on the stage in the IFFT computation. The reason is due to the way data flows during the computations within the IFFT. A more appropriate solution to the vertically folded case is a completely serialized architecture with

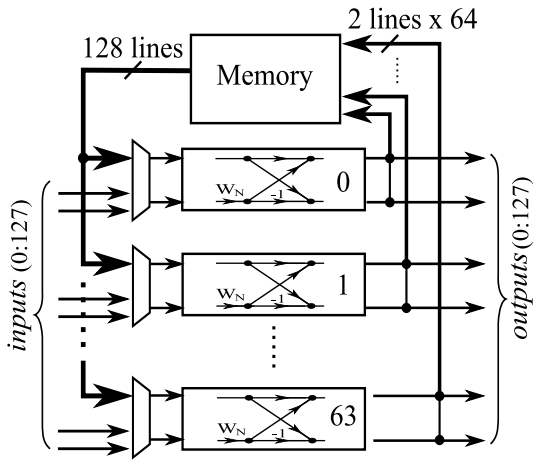

Figure 7.8: 128-point IFFT for vertically folded IOTA filter.

just one butterfly unit. However, with such an approach, the IFFT has to run at a clock that is $\frac{N}{2L}$ times faster than the IOTA filter in order to maintain the throughput between the filter and the IFFT. Though such constraints are not impossible, it certainly imposes a much harder demand on the implementation. The choice of horizontally folded architecture for the IOTA filter has a much simpler and relaxed constraint on the IFFT implementation.

For the above reasons, the horizontally folded solution is chosen for further optimizations and designing of an unified transmit/receive filter. The vertically folded architecture is discarded due to its constraints on the filter and IFFT implementation.

7.3.5 Unified filter architecture

Motivation

The previously presented architecture in Figure 7.5 has a higher area cost due to the requirement of dp-RAMs used for FIFOs. Further, the implementation of the filter at transmitter and receiver become different after time multiplexing. The difference arise because the filter coefficients make use of symmetry in the pulse shape and store fewer coefficients in the ROM. Since most radios employ transmitters and receivers as a single block, a unified architecture for the IOTA filter will result in better silicon usage and is the motivation behind the design choice. The proposed architecture is optimized to use sp-RAMs without sacrificing throughput and other performance issues by introducing reconfig-

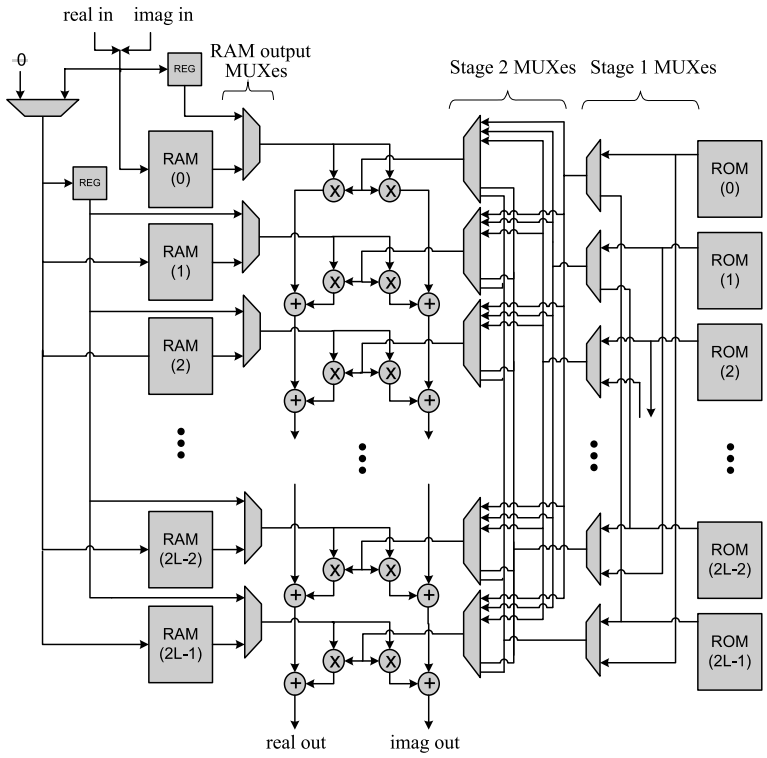

Figure 7.9: Proposed architecture implemented in ST 65nm standard cell CMOS.

urable/switching logic. It will be shown from the implementation results that the overhead is marginal.

Figure 7.9 shows the proposed architecture employing sp-RAMs and configurable as both transmit and receive IOTA filter. It consists of $2L$ RAMs and ROMs of depth N and $\left(\frac{N}{2}+1\right)$ respectively. The number of multipliers and adders required are the same as that of the horizontally folded transmit filter previously presented in Figure 7.5. In addition, it consists of $2L$ $2:1$ multiplexers denoted as *Stage 1 MUXes* and $2L$ $8:1$ multiplexers denoted as *Stage 2 MUXes*. The $2L$ $2:1$ multiplexers, to the left in Figure 7.9, at the output of the RAMs are hereby referred to as *RAM output MUXes*

Transmit/receive filter reconfigurability

Stage 1 MUXes are used to configure the filter for either *transmit* or *receive* mode. In *transmit* mode, the coefficients from ROM(0) goes to the output of the first multiplier row where the inputs come from RAM(0). In *receive* mode, the coefficients from ROM(0) are directed to the last multiplier row with inputs coming from RAM(2L-1). Accordingly, the coefficients from ROM(2L-1) is directed to the first stage of multiplier and adder for which the input and the delayed input samples come from RAM(0). In summary, in transmit mode the coefficients from each of the ROMs flow in parallel into the multipliers and adders, while in receive mode the coefficients are provided to the multipliers and adders as if the ROMs were upside down compared to that shown in the figure.

Implementing using single port RAMs

Though dp-RAMs provide the advantage of simultaneous read/write hence increasing throughput of the processing, they tend to take large area and consume a lot of power. It has been noticed that in the 65nm process technology [ST], for which the design is targeted, the dp-RAMs tend to be at least 2 times large in area compared to its single port counterparts. When using dp-RAMs as FIFOs, for every incoming input the entire data in the RAMs need to be shifted by one sample. In practice, RAMs are used as cyclic buffers thus reducing the shifts to one write operation. In steady state, when all RAMs have stored data, the number of such operations will be equal to the number of RAM blocks used, i.e., $2L$.

In the following we describe how the introduction of *Stage 2 MUXes* and *RAM output MUXes* can help in using sp-RAMs instead of dual port ones. It will also be shown in the results section that the overhead in introducing these multiplexers is acceptable compared to the area savings achieved by switching to sp-RAMs.

Stage 2 MUXes avoid multiple memory writes which were required before for every new incoming sample and will be explained below. In this approach, after a RAM block becomes full incoming samples are stored in the next adjacent RAM block. However, this results in the filter coefficients being are no longer aligned with the data. For example, when the new incoming data is written to RAM(1), these data samples should be multiplied with coefficients in ROM(0). The older data samples that were stored in RAM(0) are to be multiplied by coefficients in ROM(1). In general, the coefficients from ROM(0) have to be aligned with the incoming data samples written into newer RAM blocks. The coefficients from ROM(1) are to be aligned with the data from next most recently written RAM and so on. This problem of dynamically aligning

the coefficients to the incoming data samples is taken care of by the *Stage 2 MUXes*. With samples coming from a new OFDM symbol, the *Stage 2 MUXes* are appropriately selected in order to align the samples with the coefficients as before. This happens in a cyclic pattern and when all RAM blocks are filled by OFDM symbols, incoming symbols will replace the oldest data as they will no longer be needed to calculate outputs. This approach introduce minor overhead in the form of a controller to keep track of the data in the RAM blocks and alignment of the filter coefficients.

The *RAM output MUXes* together with an extra register overcome the drawback of simultaneous read and write operation while using sp-RAMs. In using sp-RAMs, the RAM blocks required are $2L$ instead of $2L-1$. This is because, the extra RAM stores the new incoming data samples. The register copies the same incoming sample to provide it to the arithmetic units together with the data from remaining $2L-1$ RAMs to calculate the outputs. Thus, in transitioning from using dp-RAMs to sp-RAMs the number of RAMs required are now one more than what was originally required, along with some extra logic in terms of multiplexers.

7.4 Implementation and results

The unified architecture of the transmit/receive IOTA filter is implemented using standard cell libraries from ST 65nm CMOS process [ST]. The input data and coefficient wordlengths required is evaluated from a MATLAB model of the filter. This filter model is part of a transceiver simulation chain: the multicarrier faster-than-Nyquist signaling system [DRO11]. From simulations of the entire system, the complex outputs from the IFFT block is found to be 20 bits, 10 bits each for the real and imaginary parts and the coefficients were quantized to 12 bits. This wordlength requirement is applicable to IOTA filters used in conjunction with this faster-than-Nyquist signaling system and might vary for other applications. The 12 bits for coefficient representation was due to the requirement of high precision towards the tail of the IOTA pulse.

Figure 7.10 shows the IOTA pulse from Figure 7.1 on a logarithmic scale to better visualize and compare the coefficients quantized to 8, 10, 12 and 16 bits with floating point representation. The 8 bit quantization is a poor representation of the IOTA pulse, and it results in more than half (518) of the coefficients becoming zero due to the lack of precision bits. Coefficients quantized to 10 bits provide only a small improvement compared to 8 bits with 432 zero valued coefficients. On the other hand the representation with 12 bits, though 236 coefficient values turned out to be zero, was found to be sufficient for the current application. The 16 bit representation had 58 coefficients that were zero.

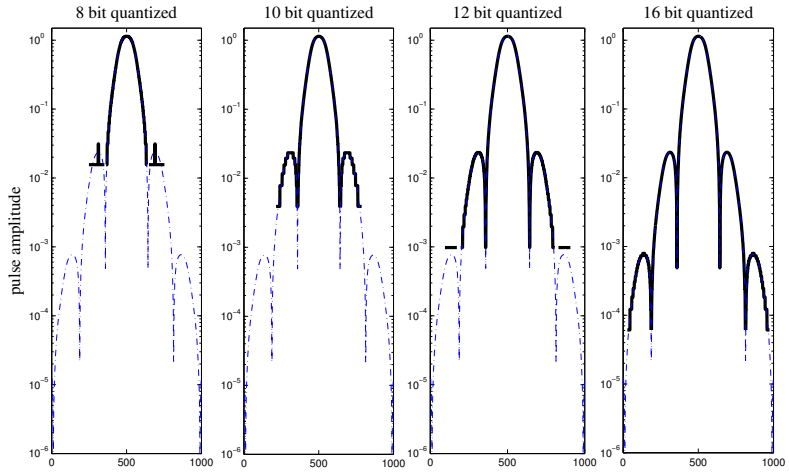

Figure 7.10: IOTA pulse representation with floating point precision (dash-dot line) in comparison with 8, 10, 12 and 16 bit quantized coefficients (thick line).

With 12 bits meeting the requirement 16 bits is not considered as it results in larger size arithmetic units. Zero valued coefficients in the filter may optimize some multiplications in the parallel implementation, but they have limited effect on the time multiplexed architectures as the multipliers are time shared amongst different filter coefficients. The requirement for high precision in the coefficient representation can be reduced by dynamically scaling the filter coefficients [BVOS04]. With this, the requirement for large wordlength multipliers, and in turn the arithmetic complexity can be reduced. With this approach, the outputs from the multipliers will have to be scaled down appropriately before summation [BVOS04]. However, this optimization has not yet been considered in the current implementation.

Resource utilization of the IOTA filter implemented in 65nm CMOS is presented in Table 7.4. The table lists the arithmetic/logic blocks used in the filter implementation along with the average unit area of these blocks in μm^2. Beside each processing block, their input wordlengths are indicated. The inputs to the multipliers are 20 bit complex valued samples and 12 bit wide coefficients producing 44 bit result in full precision. The multiplier outputs are summed up using 7 adders arranged as a 3 stage tree structure. Hence the wordlength

increases by $\lceil log_2(7) \rceil = 3$ bits each for the real and the imaginary parts to avoid overflow. Therefore, the input wordlengths of the adders are 44 and 50 respectively. ROMs that store the filter coefficients are implemented as look-up tables and are shown as a single block storing all filter coefficients. The large wordlengths are required only internally due to the high precision of the coefficients and to keep the accuracy of the calculations. The final filter outputs can be represented with a much smaller wordlength by rounding or truncating the extra precision bits from the result as presented in the beginning of this section. These wordlengths can also be reduced in applications requiring lower precision. However, this needs to be evaluated from a systems perspective on a case by case basis.

7.4.1 Resource utilization in the horizontally folded and hardware mapped architectures

The first part of Table 7.4 lists the area of the implemented filter from the architecture proposed in Figure 7.9. The synthesized design reported a maximum operating frequency of 200 MHz (4.95 ns clock period) and occupies an area of 0.11 mm^2, memories dominating with 60% of the entire filter area followed by the multipliers at 24%.

The second part of Table 7.4 lists the estimated area for a fully parallel implementation, derived from unit area of the arithmetic/logic blocks. However, in this implementation the coefficient inputs to the multipliers will be fixed, hence requiring smaller area. This has been approximated by scaling down the area of the variable multipliers by 4. Scaling down the multiplier area by 4 corresponds to a multiplier operating with half the input wordlength as before. It is to be noted that it is not possible to establish an unique value that defines the area ratio between fixed and variable multipliers since they are dependent on the value of the coefficient as well as the architecture of the multiplier unit itself. Also, when the implementation is targeted for an ASIC the speed and area constraints introduce further ambiguity. Hence, we have used the half wordlength approximation of the variable multiplier to represent a fixed multiplier which we believe is a fair approximation. Apart from the approximation of the area for the multiplier, the rest of the components would scale up by the required number of units. The parallel implementation also requires data and coefficient MUXes to be able to operate the filter in both transmit and receive modes. Registers replace the RAMs, while the requirement for ROMs does not arise in the parallel implementation. The peak operating frequency of the fully parallel filter is reported to be 1 GHz (1 ns period) and is estimated to take about 1.578 mm^2 of silicon area with multipliers and adders taking up more than 80% of the overall area.

Table 7.4: Resource utilization of the IOTA filter implemented in ST 65nm standard cell CMOS process [ST].

Arithmetic/Logic block	Unit area (in μm^2)	Proposed architecture (Figure 7.9)			Parallel architecture (Figure 7.4)			Savings factor
		No. of units	Area in μm^2	%age	No. of units	Area in μm^2	%age	
Adder ($w_1 = 44, w_2 = 50$)	1 241.88	7	8 693.16	7.94%	7×128	1 112 724.48	52.12%	128.00
Multiplier ($w = 20, w_c = 12$)	3 284.88	8	26 279.00	23.99%	8×128	840 928.00[a]	39.39%	32.00
RAM ($w = 20$, depth = 128)	8 229.60	8	65 836.80	60.10%	-			2.12[b]
ROM ($w_c = 12$, depth = 8×65)	1 906.32	1	1 906.32	1.74%	-			
Register ($w = 20$)	156.00	2	312.00	0.28%	7×128	139 776.00	6.55%	
Control logic	1 063.39	1	1 063.39	0.97%	-			
Stage1 MUXes ($w_c = 12$)	419.88	8	3 359.00	3.07%	-			
Stage2 MUXes ($w_c = 12$)	80.75	8	646.00	0.59%	2×128	20 672.00	0.97%	32.00
RAM output MUX ($w = 20$)	161.13	9	1450.13	1.32%	128	20 624.00	0.97%	14.22
Total			109 545.76	100.00%		2 134 720.00	100.00%	**19.49**

[a] Variable multiplier area is scaled by 4 to approximate a fixed multiplier. Scaling by 4 is equivalent to a multiplier of half the input wordlength.
[b] Comparison here is between RAM area in proposed arch. to the register area in parallel implementation.

7.4.2 Comparison between the proposed and parallel implementations

The time multiplexed architecture provides about 20× reduction in silicon area compared to the hardware mapped architecture. The RAMs in the proposed architecture provide more than 2× improvement in area compared to the registers in parallel implementation. The time multiplexed filter provides one output per clock cycle resulting in a peak throughput of 200M samples/s, while the full parallel implementation provides 128 outputs simultaneously resulting in a peak throughput of 128G samples/s. As previously stated, the hardware mapped filter implementation assumes a parallel implementation of the IFFT, while a pipelined implementation is sufficient for the time multiplexed architecture. As a result, the overall overhead in the parallel implementation considering both IFFT and IOTA filter implementations can be much larger than that compared with the IOTA filter alone. Furthermore, a fully parallel implementation providing a throughput of 128G samples/s is an over kill both in terms of silicon real estate as well as the speed requirement. Thus the horizontally folded time multiplexed architecture is motivated on the grounds of both throughput and area constraints.

7.4.3 Savings from using sp-RAMs

Previously, it was mentioned that in the ST 65nm process the dp-RAMs were found to be at least 2x larger in area when compared to the sp-RAMs. Assuming even 1.5x as the overall overhead between the dp- and the sp-RAMs, the dp-RAM based architecture (c.f. Figure 7.5) would require $1.5 \times 65\,836\mu\text{m}^2$ of silicon real estate for memories. Thus, the proposed architecture provide a reduction of $\approx 30\,000\mu\text{m}^2$ by using sp-RAMs. On the other hand, the architecture using sp-RAMs has the overhead from the *Stage 2 MUXes* and *RAM output MUXes* which is only about $2\,096\mu\text{m}^2$ in total. *Stage 1 MUXes* are only used to configure the filter for either transmit/receive operations and hence need not be accounted for while determining the overhead. Thus, the sp-RAM based architecture provides significant savings in area.

The layout of the implemented IOTA filter is shown in Figure 7.11 and the chip results are summarized in Table 7.5. The chip takes a total area of 0.47mm^2 including the IO pads, while the core area is 0.11mm^2. The actual core area presented in Table 7.5 is quite small compared to the overall chip size. However, in Figure 7.11 the core area seems to be much larger. This is mainly because the design is pad limited and most of the standard cell area in the core is empty. The IOTA filter not being a stand alone component, the IO pads are not of interest and can be ignored. This can be verified from Table

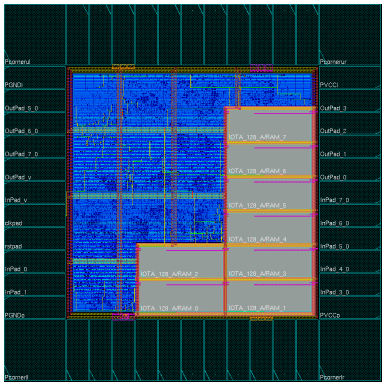

Figure 7.11: Layout of the transmit/receive IOTA filter implemented in 65nm CMOS.

Table 7.5: Summary of results of the IOTA filter implemented in ST65nm.

Total chip area	0.47 mm^2
- Core area	0.11 mm^2
Total Power	14.4 mW
- Logic	1.1 mW
- Memory	13.3 mW
Peak operating frequency	200MHz

7.4, where the RAMs occupy 60% of the filter, while the arithmetic units and other related logic form the remaining 40%. Power consumption of the chip is estimated by using Synopsys PrimeTime and the design reported a total power of 14.4mW of which 92% (13.3mW) is due to the memories.

7.4.4 IOTA filter in the FTN system

This section brings about a comparison between the IOTA filter overhead with respect to the FTN system in which it is employed [DRO11]. Table 7.6 summarizes the area of the IOTA filter as well as the component blocks that form the FTN iterative decoder. Comparing the area of the IOTA filter to that of the memory optimized implementation of the FTN decoder it can be deduced that the IOTA filter takes up 28% of the area of the entire FTN decoder or about 44% of the inner decoder. The inner decoder is the component decoder

Table 7.6: Area comparison between the IOTA filter and key components of the FTN iterative decoder in 65nm CMOS.

Blocks	Area in μm^2	Memory
Inner Decoder	248 519	4.86 kB
Outer Decoder	54 502	4.92 kB
Π and Π^{-1}	85 100	3.93 kB
Global FSM	3 086	-
FTN decoder (total)	391 207	13.71 kB
IOTA filter	109 545	2.50 kB

responsible for decoding of FTN modulated signals and hence is interesting for comparison. Relative to the FTN decoder, the memory requirement in the IOTA filter is only about 18%. Further this memory overhead is already included in the reported overall area overhead of 28%. It has also to be noted that with this acceptable overhead in area, the same filter architecture can be used for both transmit and receive filtering. With IOTA providing a two fold advantage of eliminating the cyclic prefix (usually about 25% of the transmitted OFDM symbol) as well as reducing the number of operations per transmitted symbol in the FTN system [DRAO09], the overall area overhead of 28% is well justified.

7.5 Summary

The IOTA pulse shaping is presented in this chapter to bring about a comparison with the FTN decoder and also show that the overhead due to the pulse shaping filter can be kept moderate. The use of IOTA pulse has aided in complexity reduction of the processing blocks involved in FTN signaling and hence an efficient implementation of the IOTA filter itself is important. Different architectural flavors of the filter is studied with respect to throughput and area requirements considering the fact that the IOTA filter is a part of the larger FTN multicarrier system. Finally, an unified architecture for IOTA transmit/receive pulse shaping filters is proposed and implemented. A comparison is brought out between the hardware complexity of the IOTA filter and the FTN decoder.

Conclusion

This thesis has dealt with both algorithmic and architectural aspects of FTN signaling for hardware implementation. On an algorithmic level, lower complexity is achieved by considering fewer operations per FTN symbol; reuse of processing blocks (FTN mapper, MF) to realize different functionality; operating at sub-optimal points to reduce LUT sizes. The receiver performance with FTN signaling in AWGN channels is shown to approach that of an orthogonal system. This indicates improvement in bandwidth usage achieved by extra processing in the transmitter and receiver. The receiver is also analyzed for its performance in frequency selective channels and adaptive FTN signaling is used to our advantage to maximize data rate.

On the architectural front, LUT based implementation of the FTN mapper in the transmitter helped in transmitting information at varied bandwidth efficiencies including orthogonal signaling. In the receiver, the FTN iterative decoder architecture was designed by reusing the processing blocks and keeping hardware overhead moderate. Hardware overhead analysis comparing the FTN specific inner decoder and the outer decoder, a popular channel decoding approach, showed that the complexity with FTN signaling is acceptable. The iterative decoder architecture implemented in an ASIC using a state-of-the-art 65 nm CMOS process showed that the performance deviation between the decoder in silicon and the MATLAB benchmark was within 0.5 dB. It was also demonstrated in silicon that the FTN decoder is capable of operating in both FTN and orthogonal signaling modes.

In conclusion, hardware aspects and feasibility of FTN signaling for multi-carrier systems have been explored. This work has established that bandwidth efficiency can be improved with FTN signaling and moderate increase in processing complexity, proving the usability of FTN for practical applications.

Future directions

This chapter suggests extensions that can be undertaken to further evaluate FTN signaling in multicarrier systems. The future directions are classified into theoretic/algorithmic extensions, architectural extensions and extensions in other parts of the baseband system.

Theoretic/algorithmic extensions

On the theoretical front, the inner decoder responsible for decoding of FTN modulated signals can be improved with another decoding algorithm. Aspects such as parallelism in the algorithm that can improve throughput is one of the key issues that needs to be explored. A much stronger outer code such as an LDPC can be used in place of the current $(7, 5)$ convolutional code to improve decoding performance. LDPCs being inherently iterative in nature, can help in reducing the overall number of iterations during decoding. Some preliminary studies carried out indicated that, the LDPC in itself can be restricted to about 3 (internal) iterations while the message passing between the outer and the inner decoder can be confined to about $3 - 4$ (global) iterations. As indicated, the results being preliminary they are not presented in any of the main sections in this thesis and needs further investigation. Also, the inclusion of LDPC as the outer code and a better inner decoder algorithm with parallelism together helps in providing enhanced data throughput rates.

Architectural extensions

The FTN decoder chip presented in this work decodes symbols received over an AWGN channel. A straightforward extension is to incorporate the effects of fading which can be achieved without major architectural changes. In fact, only the inner decoder requires modifications as it accounts for the effects of fading by considering the channel coefficients. If the decoder hardware has to incorporate adaptive FTN signaling discussed in Chapter 3, the inner decoder has to be modified such that each sub-band is treated as a smaller and independent block of information. This requires multiple instantiations of the mapper and the MF which in turn can bring down the decoding latency in the inner decoder as the processing of the information in the sub-bands are performed in parallel.

The FTN system can be seen as a serial concatenation of two codes and in this setup, the inner decoder remains idle while the outer one is decoding and vice versa. From an hardware implementation point of view, this can be used to either reduce power consumption or double the data throughput. Power savings can be achieved by shutting down the decoder that is idle during a decoding cycle. Throughput improvement requires duplication of the interleaver and de-interleaver. By doing so, two blocks of information can be decoded simultaneously with each decoder operating on separate blocks of information at any time. Finally, the entire FTN decoder implementation can be overhauled with inclusion of proposed algorithmic extensions from the previous section.

Extensions in other parts of the system

The introduction of FTN signaling in multicarrier systems directly or indirectly influences the way other processing blocks are to be realized. This needs to be investigated and studied in greater detail. A couple of these aspects that are affected due to FTN signaling and might need immediate attention are briefly discussed here.

Channel estimation

Conventionally, multicarrier systems use pilot symbols placed at certain sub-carrier positions in place of information symbols so that the channel can be estimated using these pilots. In the case of an FTN system this is not straight forward due to the presence of FTN mapper preceding the IFFT. It can be recollected that the FTN mapper projects information from several FTN symbols on a particular sub-carrier. Due to this, introducing pilots become complicated or may result in loss of information.

One solution is to transmit pilot frames instead of embedding them between data symbols. Using pilot frames tend to be disadvantageous if the channel is changing rapidly. Alternatively, pilots can be used to replace data at chosen sub-carrier positions. On the outset, it seems that such an approach will result in loss of information as these sub-carrier positions will no longer contribute to FTN symbol reconstruction. Due to the mapping operation only a fraction of the actual FTN symbol information would be present at the positions where the pilots are introduced, indicating that the degradation would not be severe. However, this tradeoff needs to be evaluated to determine the impact of ignoring the information. Elaborate studies to find alternative solutions for pilot positioning without sacrificing decoder performance is needed.

Peak to Average Power Ratio (PAPR)

High peak to average power ratio is generally an issue in OFDM systems. In FTN based systems, this issue can be more aggravated due to the accumulation of the projection coefficients. High PAPR puts a higher demand on the power amplifiers for linearity and dynamic range. This can introduce bigger challenges in power amplifier (PA) design when FTN signaling is employed in multicarrier systems. At the same time, FTN systems achieve higher bandwidth efficiency. Hence, in order to evaluate the design challenges in FTN systems, the PA requirement has to be considered to an equivalent higher order modulation scheme.

Bibliography

[3GP04] 3GPP. Feasibility Study for Orthogonal Frequency Division Multiplexing (OFDM) for UTRAN enhancement. Technical Report TR 25.892 V6.0.0, 3rd Generation Partnership Project, Jun. 2004.

[3GP07] 3GPP. *Technical specification TS 36.212: multiplexing and channel coding (release 8)*. Dec. 2007.

[3GP10] 3GPP. *Technical specification TS 25.306:UE Radio access capabilities*. Jun. 2010.

[Ala01] M. Alard. Construction of a multicarrier signal. *US Patent No. 6,278,686*, Aug. 2001.

[BBK04] M. Batariere, K. Baum, and T.P. Krauss. Cyclic prefix length analysis for 4G OFDM systems. In *IEEE Vehicular Technology Conference (VTC-Fall)*, volume 1, pages 543–547, Sep. 2004.

[BCJR74] L. Bahl, J. Cocke, F. Jelinek, and J. Raviv. Optimal decoding of linear codes for minimizing symbol error rate. *IEEE Transactions on Information Theory*, 20(2):284–287, Mar. 1974.

[BD74] M. Bellanger and J. Daguet. TDM-FDM Transmultiplexer: Digital Polyphase and FFT. *IEEE Trans. on Communications*, 22(9):1199–1205, Sep. 1974.

[BDH99] H. Bölcskei, P. Duhamel, and R. Hleiss. Design of pulse shaping OFDM/OQAM systems for high data-rate transmission over wireless channels. In *Proc. IEEE International Conference on Communications (ICC)*, pages 559–564, Jun. 1999.

[BDH03] H. Bölcskei, P. Duhamel, and R. Hleiss. Orthogonalization of OFDM/OQAM pulse shaping filters using the discrete Zak transform. *Signal Processing (EURASIP)*, 83(7):1379–1391, Jul. 2003.

[BDMP98] S. Benedetto, D. Divsalar, G. Montorsi, and F. Pollara. Serial concatenation of interleaved codes: Performance analysis, design, and iterative decoding. *IEEE Trans. on Information Theory*, 44(3):909–926, May. 1998.

[BER] BEREC. http://erg.eu.int.

[BFC09] A. Barbieri, D. Fertonani, and G. Colavolpe. Time-frequency packing for linear modulations: Spectral efficiency and practical detection schemes. *IEEE Trans. on Communications*, 57(10):2951–2959, Oct. 2009.

[Bol03] H. Bolcskei. *Orthogonal Frequency Division Multiplexing based on offset QAM*, pages 351–352. Birkhauser, 2003.

[BS91] D. H. Bailey and P. N. Swarztrauber. The fractional Fourier transform and applications. *SIAM Review*, 33(3):389–404, 1991.

[BVOS04] H. Bruce, R. Veljanovski, V. Owall, and J. Singh. Power optimization of a reconfigurable FIR-filter. In *IEEE Workshop on Signal Processing Systems*, pages 321–324, Oct. 2004.

[Cav72] J. Cavers. Variable-rate transmission for Rayleigh fading channels. *IEEE Transactions on Communications*, 20(1):15–22, Feb. 1972.

[Cha66] R. W. Chang. High-speed multichannel data transmission with bandlimited orthogonal signals. *Bell System Technical Journal*, 45:1775–1796, Dec 1966.

[CKRD10] A. Chorti, I. Kanaras, M.R.D. Rodrigues, and I. Darwazeh. Joint channel equalization and detection of Spectrally Efficient FDM signals. In *International Symposium on Personal Indoor and Mobile Radio Communications (PIMRC)*, pages 177–182, Sep. 2010.

[CLJ06] P. M. Combelles, D. M. LaCroix, and A. M. Jalali. Multicarrier modulation using weighted prototype functions. *US Patent No. 7,099,396*, Aug. 2006.

[DAK+09] D. Dasalukunte, K. Ananthanarayanan, M. Kandasamy, F. Rusek, and V. Öwall. Hardware implementation of mapper for faster-than-Nyquist signaling transmitter. In *Proc. IEEE NORCHIP*, Nov. 2009.

[Den09] P. W. Dent. Method and apparatus for communication with root-Nyquist self-transform pulse shapes. *US Patent No. 2009/0003472 A1*, Jan. 2009.

[DPS08] E. Dahlman, S. Parkvall, and J. Skold. *3G Evolution: HSPA and LTE for Mobile Broadband*. Academic Press, 2 edition, 2008.

[DRAO09] D. Dasalukunte, F. Rusek, J. B. Anderson, and V. Owall. Transmitter architecture for faster-than-Nyquist signaling systems. In *IEEE International Symposium on Circuits and Systems(ISCAS)*, pages 1028–1031, May. 2009.

[DRO10] D. Dasalukunte, F. Rusek, and V. Öwall. An iterative decoder for multicarrier faster-than-Nyquist signaling systems. In *Proc. IEEE International Conference on Communications*, May. 2010.

[DRO11] D. Dasalukunte, F. Rusek, and V. Öwall. Multicarrier faster-than-Nyquist signaling transceivers: Hardware architecture and performance analysis. *IEEE Transactions on Circuits and Systems-I*, 58(4):827–838, Apr. 2011.

[ESvdB+98] O. Edfors, M. Sandell, J.-J. van de Beek, S.K. Wilson, and P.O. Borjesson. OFDM channel estimation by singular value decomposition. *IEEE Transactions on Communications*, 46(7):931 –939, Jul. 1998.

[FCC] FCC. www.fcc.gov.

[Fli94] N. J. Fliege. *Multirate Digital Signal Processing*. Wiley, 1994.

[Gal63] R. G. Gallager. *Low Density Parity Check Codes*. Monograph, MIT press, 1963.

[GC97] A. J. Goldsmith and S. G. Chua. Variable-rate variable-power MQAM for fading channels. *IEEE Transactions on Communications*, 45(10):1218 –1230, Oct. 1997.

[GSM] GSM. www.etsi.org/website/technologies/gsm.aspx.

[Hir81] B. Hirosaki. An orthogonally multiplexed QAM system using the discrete Fourier transform. *IEEE Trans. on Communications*, 29(7):982–989, Jul. 1981.

[HT98] S. He and M. Torkelson. Designing pipeline FFT processor for OFDM (de)modulation. In *1998 URSI International Symposium on Signals, Systems, and Electronics*, pages 257–262, Sep 1998.

[HT04] M. Hamamura and S. Tachikawa. Bandwidth efficiency improvement for multi-carrier systems. In *IEEE International Symposium on Personal, Indoor and Mobile Radio Communications*, volume 1, pages 48–52, Sep. 2004.

[HZ09] F. M. Han and X. D. Zhang. Wireless multicarrier digital transmission via Weyl-Heisenberg frames over time-frequency dispersive channels. *IEEE Trans. on Communications*, 57:1721–1733, Jun. 2009.

[ID11a] S. Isam and I. Darwazeh. Design and performance assessment of fixed complexity spectrally efficient FDM receivers. In *IEEE Vehicular Technology Conference (VTC Spring)*, pages 1–5, May. 2011.

[ID11b] S. Isam and I. Darwazeh. Peak to average power ratio reduction in spectrally efficient FDM systems. In 18^{th} *International Conference on Telecommunications (ICT)*, pages 363–368, May. 2011.

[Int] Intel. www.intel.com.

[ITR] ITRS. ITRS update 2010.

[JL03] J. P. Javaudin and D. Lacroix. Technical description of the OFDM/IOTA modulation. Technical Report R1-03-168, France Telecom R&D, 2003.

[KB10] Y.J.D. Kim and J. Bajcsy. On spectrum broadening of pre-coded faster-than-Nyquist signaling. In *Proc. IEEE Vehicular Technology Conference (VTC) Fall*, Sep. 2010.

[KCRD09] I. Kanaras, A. Chorti, M.R.D. Rodrigues, and I. Darwazeh. Spectrally efficient fdm signals: Bandwidth gain at the expense of receiver complexity. In *IEEE International Conference on Communications*, pages 1–6, Jun. 2009.

[KÅ09] J. Kåredal. *Measurement-Based Modeling of Wireless Propagation Channels - MIMO and UWB*. PhD thesis, Dept. of Electrical and Information Technology, Lund University, 2009.

[LC04] S. Lin and D. J. Costello. *Error Control Coding*. Prentice-Hall, Inc., 2 edition, 2004.

[Lee01] I. Lee. The effect of a precoder on serially concatenated coding systems with an ISI channel. *IEEE Transactions on Communications*, 49(7):1168–1175, Jul. 2001.

[LFAB95] B. Le Floch, M. Alard, and C. Berrou. Coded Orthogonal Frequency Division multiplex. *Proceedings of the IEEE*, 83(6):982–996, Jun. 1995.

[LG03] A. D. Liveris and C. N. Georghiades. Exploiting faster-than-Nyquist signaling. *IEEE Trans. on Communications*, 51(9):1502–1511, Sep. 2003.

[LG11] F. F. Lanas and P. P. Gomez. Global communications newsletter. *IEEE Communications Magazine*, 49(7), Jul 2011.

[LGA01] D. Lacroix, N. Goudard, and M. Alard. OFDM with guard interval versus OFDM/offsetQAM for high data rate UMTS downlink transmission. In *Proc. IEEE Vehicular Technology Conference (VTC) Fall*, volume 4, pages 2682–2686, Oct. 2001.

[LL04] J. H. Lee and Y. H. Lee. Design of multiple MMSE sub-equalizers for faster-than-Nyquist-rate transmission. *IEEE Trans. on Communications*, 52(8):1257–1264, Aug. 2004.

[Maz75] J. E. Mazo. Faster-than-Nyquist signaling. *Bell System Technical Journal*, 54(8):1451–1462, Oct. 1975.

[MB00] G. Montorsi and S. Benedetto. Design of fixed-point iterative decoders for concatenated codes with interleavers. In *Proc. IEEE Global Telecommunications Conference (GLOBECOM)*, pages 801–806, 2000.

[MJ05] M. Muck and J. P. Javaudin. Advanced OFDM modulators considered in the IST-WINNER framework for future wireless systems. In 14^{th} *IST Mobile and Wireless Communications Summit conference*, 2005.

[Mol05] A. Molisch. *Wireless Communication*. Wiley., 2005.

[Moo65] G. E. Moore. Cramming more components onto integrated circuits. *Electronics*, 38(8), Aor. 1965.

[MRB10] P. Meinerzhagen, C. Roth, and A. Burg. Towards generic low-power area-efficient standard cell based memory architectures. In *IEEE International Midwest Symposium on Circuits and Systems (MWSCAS)*, pages 129–132, Aug. 2010.

[MS10] M. McGuire and M. Sima. Discrete time faster-than-Nyquist signalling. In *Proc. of IEEE Global Telecommunication Conference (GLOBECOM)*, Dec. 2010.

[MWW00] H. Michel, A. Worm, and N. Wehn. Influence of quantization on the bit-error performance of turbo-decoders. In *Proc. IEEE Vehicular Technology Conference (VTC) Spring*, pages 581–585, May. 2000.

[Nyq28] H. Nyquist. Certain Topics in Telegraph Transmission Theory. *Transactions of the American Institute of Electrical Engineers*, 47(2):617–644, Apr. 1928.

[Par00] B. Parhami. *Computer Arithmetic: Algorithms and Hardware designs*. Oxford Univeristy Press, 2000.

[PD11] M.R. Perrett and I. Darwazeh. Flexible hardware architecture of SEFDM transmitters with real-time non-orthogonal adjustment. In 18$^{\text{th}}$ *International Conference on Telecommunications (ICT)*, pages 369–374, May. 2011.

[PM04] J. G. Proakis and D. G. Manolakis. *Digital Signal Processing: Principles, Algorithms, and Applications*. Prentice-Hall, Inc., 3 edition, 2004.

[PS08] J. G. Proakis and M. Salehi. *Digital Communications*. McGraw Hill, 5 edition, 2008.

[RA09a] F. Rusek and J. B. Anderson. Constrained capacities for faster-than-Nyquist signaling. *IEEE Trans. on Information Theory*, 55(2):764–775, Feb. 2009.

[RA09b] F. Rusek and J. B. Anderson. Multistream faster-than-Nyquist signaling. *IEEE Trans. on Communications*, 57(5):1329–1340, May. 2009.

[RH97] P.K. Remvik and N. Holte. Carrier frequency offset robustness for OFDM systems with different pulse shaping filters. In *IEEE Global Telecommunications Conference (GLOBECOM)*, volume 1, pages 11–15, Nov. 1997.

[RSAA98] M.C. Reed, C.B. Schlegel, P.D. Alexander, and J.A. Asenstorfer. Iterative multiuser detection for CDMA with FEC: near-single-user performance. *IEEE Trans. on Communications*, 46:1693–1699, Dec. 1998.

[Rus07] F. Rusek. *Partial response and faster-than-Nyquist signaling*. PhD thesis, Dept. of Electrical and Information Technology, Lund University, 2007.

[RVH95] P. Robertson, E. Villebrun, and P. Hoeher. A comparison of optimal and sub-optimal MAP decoding algorithms operating in the log domain. In *IEEE International Conference on Communications (ICC)*, volume 2, pages 1009–1013, Jun. 1995.

[Sal67] B. Saltzberg. Performance of an efficient parallel data transmission system. *IEEE Trans. on Communication Technology*, 15(6):805–811, Dec. 1967.

[SCZP07] C. Sahnine, D. Callonnec, N. Zergainoh, and F. Petrot. Efficient design approach and advanced architectures for universal OFDM systems. In *Ph.D. Research in Microelectronics and Electronics Conference (PRIME)*, pages 33 –36, Jul. 2007.

[SSL02] P. Siohan, C. Siclet, and N. Lacaille. Analysis and design of OFDM/OQAM systems based on filterbank theory. *IEEE Trans. on Signal Processing*, 50(5):1170–1183, May. 2002.

[SSP06] C. Siclet, P. Siohan, and D. Pinchon. Perfect reconstruction conditions and design of oversampled DFT-modulated transmultiplexers. *EURASIP Journal on Applied Signal Processing*, Jan. 2006.

[ST] ST. 65nm CMOS process.

[SZCP07] C. Sahnine, N. Zergainoh, D. Callonnec, and F. Petrot. Towards a high-throughput and low power reconfigurable architecture of advanced OFDM modulator for software-defined radio systems. In *IEEE Northeast Workshop of Circuits and Systems (NEWCAS)*, pages 1205–1208, 2007.

[TI] TI. www.ti.com.

[TRA] TRAI. www.trai.gov.in.

[UMC] UMC. 130nm CMOS process.

[Ver98] S Verdu. *Multiuser Detection*. Cambridge University Press, 1998.

[Vol59] J. E. Volder. The CORDIC trigonometric computing technique. *IRE Trans. on Electronic Computers*, EC-8(3):330–334, Sep. 1959.

[WD84] E.H. Wold and A.M. Despain. Pipeline and Parallel-Pipeline FFT Processors for VLSI Implementations. *IEEE Trans. on Computers*, C-33(5):414–426, May. 1984.

[WHW00] A. Worm, P. Hoeher, and N. Wehn. Turbo-decoding without SNR estimation. *IEEE Communications Letters*, 4:193–195, Jun. 2000.

[Wik] Wikipedia. http://en.wikipedia.org/wiki/3g_spectrum_auction_india.

[WP99] X. Wang and H. V. Poor. Iterative (turbo) soft interference cancellation and decoding for coded CDMA. *IEEE Trans. on Communications*, 47:1046–1061, Jul. 1999.

[WPID11] P. N. Whatmough, M. R. Perrett, S. Isam, and I. Darwazeh. VLSI architecture for a reconfigurable Spectrally Efficient FDM baseband transmitter. In *Proc. IEEE International Symposium on Circuits and Systems (ISCAS)*, pages 1688 –1691, May. 2011.

[Xila] Xilinx. CORE generator.

[Xilb] Xilinx. Virtex-II Pro datasheet (xc2vp30).

[YC10] Y. G. Yoo and J. H. Cho. Asymptotic Optimality of Binary Faster-than-Nyquist Signaling. *IEEE Communications Letters*, 14(9):788 –790, Sep. 2010.