Real-Time BaseBand Processing Architectures for Wireless MIMO Communication Systems

Thanasis Karachalios

Department of Physics

National and Kapodistrian University of Athens

This dissertation is submitted for the degree of

Doctor of Philosophy

National and Kapodistrian University of Athens 2017

PhD Dissertation

Real-Time BaseBand Processing Architectures for Wireless MIMO Communication Systems

Thanasis Karachalios (R.N.: 2008-525)

Three-Member Advisory Committee:

Manolis Tsilis (Assist. Prof. NKUA) [Supervisor]
Dionysis Reisis (Assoc. Prof. NKUA)
Aggelos Katsaggelos (Prof. NKUA)

Seven-Member Examination Committee:

Manolis Tsilis,
Assist. Prof. NKUA

Elias Manolakos,
Prof. NKUA

Dionysis Reisis,
Assoc. Prof. NKUA

Dimitrios Soudris,
Assoc. Prof. NTUA

Aggelos Katsaggelos,
Prof. NKUA

Konstantinos Nikitopoulos,
Assist. Prof. Surrey Univ. (UK)

Andreas Polydoros,
Prof. NKUA

Examination Date: September 4, 2017

Διδακτορική Διατριβή

Αρχιτεκτονικές Η/Υ Πραγματικού Χρόνου για Επεξεργασία Δεδομένων Βασικής Ζώνης σε Ασύρματα Τηλεπικοινωνιακά Συστήματα Πολλαπλών Κεραιών

Αθανάσιος Καραχάλιος (Α.Μ.: 2008-525)

Τριμελής Συμβουλευτική Επιτροπή:

Τσίλης Μανώλης (Επικ. Καθ. ΕΚΠΑ) [Επιβλέπων]
Ρεΐσης Διονύσης (Αν. Καθ. ΕΚΠΑ)
Κατσάγγελος Άγγελος (Καθ. ΕΚΠΑ)

Επταμελής Εξεταστική Επιτροπή:

Τσίλης Μανώλης,
Επικ. Καθ. ΕΚΠΑ

Μανωλάκος Ηλίας,
Καθ. ΕΚΠΑ

Ρεΐσης Διονύσης,
Αν. Καθ. ΕΚΠΑ

Σούντρης Δημήτριος,
Αν. Καθ. ΕΜΠ

Κατσάγγελος Άγγελος,
Καθ. ΕΚΠΑ

Νικιτόπουλος Κωνσταντίνος,
Επικ. Καθ. Παν. Surrey (UK)

Ανδρέας Πολύδωρος,
Καθ. ΕΚΠΑ

Ημερομηνία Εξέτασης: 4 Σεπτεμβρίου, 2017

To my parents and my teachers ...

Ευχαριστίες

Θα ήθελα να ευχαριστήσω τον κ. Ρείση για τη πολύχρονη και πολύτιμη καθοδήγησή, βοήθεια και γνώση καθώς επίσης και την αμέριστη υποστήριξή του από τα πρώτα μου χρόνια στο τμήμα Φυσικής. Επιπλέον, τον κ. Τσίλη και τον κ. Κατσάγγελο καθώς και τα υπόλοιπα μέλη της εξεταστικής επιτροπής για τις πολύτιμες και εποικοδομητικές επισημάνσεις τους. Τους συναδέλφους μου στο εργ. "Μελέτης και Ανάπτυξης Ψηφιακών Συστημάτων" του τμ. Φυσικής για την πολυετή επιτυχημένη συνεργασία μας και τις πολύωρες συζητήσεις μας...

Τέλος θα ήθελα να ευχαριστήσω την οικογένειά μου και τη Μαριάννα για την υποστήριξη και την υπομονή τόσων χρόνων...

Abstract

Multiple-Input Multiple-Output (MIMO) communication is a rapidly developing wireless technology which promises increased data-rates and link reliability with mobility and high quality-of-service (QoS) for multiple users. Several antennas at both transmitter and receiver increase the channel capacity with efficient bandwidth utilization, due to multiple data transmission over the same frequency bands. MIMO technologies, in combination with the *Orthogonal Frequency Division Multiplexing (OFDM)* modulation, have been adopted by many wireless standards such as WiFi (IEEE-802.11n/ac), Long Term Evolution (3GPP-LTE) and WiMAX (IEEE-802.16e) and they expected to play a key role in the upcoming WiFi (IEEE-802.11ax) standard and in the fifth generation (5G) mobile phone systems. Nevertheless, the advantages provided by MIMO technologies come at the expense of a substantial increase in the complexity mainly of the receiver, but in some cases also in the transmitter side, which has a major impact on the implementation cost and power consumption of MIMO-OFDM systems.

The modern integrated circuits has increased transistor capacity due to the advanced submicron technology, which provide the flexibility to design a single chip to support multiple protocols, with a *Software Defined Radio (SDR)* architecture. In a SDR system the computational intensive and time-critical processes are mapped to hardware accelerator units, which should have various operational modes and run-time reconfigurations to support the system requirements of multiple protocols. Furthermore, these hardware units should have increased scalability, low complexity and implementation cost and reduced power consumption to support SDR systems running on battery. Therefore, the design of low-complexity and power optimal SDR and MIMO architectures is an important issue, which is tackled throughout this thesis.

The first part of this dissertation presents the state of the art FFT architectures for OFDM

systems with multiple data streams (MIMO-OFDM). A detailed analysis in terms of complexity, scalability, implementation cost and power consumption is presented and the possibility of SDR support is investigated, for these architectures. A novel memory-based FFT architecture is proposed with increased scalability and support for operation on advanced SDR systems. The efficient conflict-free addressing scheme reduces the complexity of the interconnection network and the memory requirements of the FFT processor, resulting in a low implementation cost and power consumption, even in the case of continuous-flow operation. The reconfigurable architecture can be tailored to match any SDR system requirements, while a scheduling mechanism can be used to optimize the processing latency of the FFT processor, based on run-time parameters.

In the second part of this thesis MIMO detection architectures are investigated, in terms of complexity and error-rate performance. The computationally intensive task of tree node enumeration on sphere decoders is analyzed and the state of the art algorithms are presented, for the cases of hard decision detection and detection with the use of soft information. An advanced enumeration technique is proposed, for hard or soft sphere decoders, which can guarantee the optimal detection for all scenarios and channel conditions. The proposed method is based on a predefined visiting order, a single distance calculation unit and a tuned pruning metric, which increases the number of visiting nodes but with low computational requirements per node and reduced total complexity, for the detection process. The architecture of the proposed method is presented and the ASIC and FPGA implementation is compared with implementations of the state of the art optimal enumeration algorithms. The efficient architecture of the proposed technique leads to reduced implementation cost and power consumption, resulting in its potential use in more complex MIMO-OFDM systems.

SUBJECT AREA: Real-Time and Parallel Baseband Architectures, MIMO-OFDM Baseband Processing

KEYWORDS: Real-Time MIMO Baseband Architectures, FFT processor, Parallel FFT Architectures, MIMO Detection, Sphere Decoder

Περίληψη

Η χρήση πολλαπλών κεραιών σε ασύρματα τηλεπικοινωνιακά συστήματα είναι μια άκρως αναπτυσσόμενη τεχνολογία η οποία υπόσχεται αυξημένη ταχύτητα μεταφοράς δεδομένων και αξιόπιστη διασύνδεση, για μεγαλύτερο αριθμό χρηστών με βελτιωμένη ποιότητα υπηρεσιών. Η πιο αποδοτική χρήση του εύρους ζώνης οδηγεί σε αυξημένη χωρητικότητα καναλιού, με δεδομένη τη πολλαπλή χρήση των ίδιων συχνοτήτων, για τη μετάδοση δεδομένων. Η τεχνολογία πολλαπλών κεραιών μετάδοσης και λήψης (MIMO), σε συνδυασμό με τη διαμόρφωση ορθογώνιας πολυπλεξίας συχνότητας (OFDM) έχει υιοθετηθεί από πολλά σύγχρονα πρωτόκολλα ασύρματων επικοινωνιών, όπως το WiFi (IEEE-802.11n/ac), το LTE (3GPP-LTE) και το WiMAX (IEEE-802.16e) και αναμένεται να διαδραματίσει καθοριστικό ρόλο στα συστήματα κινητών επικοινωνιών πέμπτης γενιάς (5G), καθώς και στο επόμενο πρωτόκολλο WiFi (IEEE-802.11ax). Παρά τα πλεονεκτήματα της χρήσης πολλαπλών κεραιών εκπομπής και λήψης, η συγκεκριμένη τεχνολογία αυξάνει σημαντικά τη πολυπλοκότητα του δέκτη και σε αρκετές περιπτώσεις και του πομπού, με αποτέλεσμα την αύξηση του κόστους εφαρμογής και της κατανάλωσης ενέργειας στα συστήματα MIMO-OFDM.

Η αυξημένη πυκνότητα σε τρανζίστορ, των σύγχρονων ολοκληρωμένων κυκλωμάτων, λόγω της τεχνολογίας λιθογραφίας με διαστάσεις μικρότερες του μικρόμετρου, παρέχει τη δυνατότητα υποστήριξης πολλαπλών πρωτοκόλλων ασύρματης επικοινωνίας σε ένα μόνο ολοκληρωμένο κύκλωμα, με τη χρήση αρχιτεκτονικών καθορισμένων σε λογισμικό (Software Defined Ratio - SDR). Σε τέτοιες αρχιτεκτονικές, οι υπομονάδες με μεγάλη πολυπλοκότητα ή με κρίσιμη καθυστέρηση για το σύστημα υλοποιούνται σε υλικό με τη μορφή μονάδων επιτάχυνσης. Τέτοιες μονάδες θα πρέπει να διαθέτουν πολλαπλούς τρόπους λειτουργίας και να είναι σε θέση να αναδιαμορφωθούν σε πραγματικό χρόνο, για να μπορούν να υποστηρίξουν τις απαιτήσεις πολλαπλών πρωτοκόλλων επικοινωνίας.

Επιπλέον, οι μονάδες αυτές θα πρέπει να έχουν αυξημένη επεκτασιμότητα, μειωμένη πολυπλοκότητα και κόστος εφαρμογής και μειωμένη κατανάλωση ενέργειας για να μπορούν να υποστηρίξουν συστήματα SDR με χρήση μπαταρίας. Επομένως, η σχεδίαση μονάδων χαμηλής πολυπλοκότητας με βελτιστοποιημένη κατανάλωση ενέργειας για συστήματα πολλαπλών κεραιών μετάδοσης και λήψης (MIMO) και συστήματα SDR είναι σημαντική και θα αντιμετωπιστεί στη παρούσα διατριβή.

Στο πρώτο τμήμα της εργασίας παρουσιάζονται σύγχρονες αρχιτεκτονικές επεξεργαστή ταχυ-μετασχηματισμού Fourier (FFT), για συστήματα ορθογώνιας πολυπλεξίας συχνότητας (OFDM) με πολλαπλές ροές δεδομένων (υποστήριξη συστημάτων MIMO-OFDM). Η λεπτομερής ανάλυση των συγκεκριμένων αρχιτεκτονικών αφορά την πολυπλοκότητα, την επεκτασιμότητα, το κόστος εφαρμογής και τη κατανάλωση ενέργειας, καθώς επίσης ερευνάται και η πιθανότητα χρήσης των συγκεκριμένων αρχιτεκτονικών σε συστήματα SDR. Προτείνεται μια πρωτοποριακή αρχιτεκτονική επεξεργαστή FFT, βασισμένη σε μνήμη, με αυξημένη επεκτασιμότητα και δυνατότητα υποστήριξης πολύπλοκων συστημάτων MIMO-OFDM και SDR. Χρησιμοποιώντας ένα αποδοτικό σύστημα διευθυνσιοδότησης της μνήμης, με αποφυγή συγκρούσεων, είναι δυνατή η μείωση της πολυπλοκότητας του δικτύου διασύνδεσης και των απαιτήσεων του επεξεργαστή σε μνήμη, με αποτέλεσμα τη μείωση του κόστους εφαρμογής και της κατανάλωσης ενέργειας, ακόμα και στη περίπτωση λειτουργίας συνεχούς ροής δεδομένων. Η αναδιαμορφώμενη αρχιτεκτονική μπορεί να προσαρμοστεί με βάση τις απαιτήσεις ακόμα και των πιο πολύπλοκων συστημάτων SDR, ενώ ο μηχανισμός χρονοπρογραμματισμού του επεξεργαστή μπορεί να χρησιμοποιηθεί για τη βελτιστοποίηση της καθυστέρησης της επεξεργασίας με τη χρήση παραμέτρων, που μπορούν να αλλάζουν κατά τη διάρκεια της λειτουργίας.

Στο δεύτερο τμήμα της διατριβής ερευνώνται αρχιτεκτονικές μονάδων αποκωδικοποίησης σήματος σε συστήματα πολλαπλών κεραιών εκπομπής και λήψης, με βάση τη πολυπλοκότητα και την απόδοσή τους. Αναλύεται η πολύπλοκη διεργασία της απαρίθμησης των κόμβων του δέντρου, σε αρχιτεκτονικές αποκωδικοποιητή σφαίρας και παρουσιάζονται σύγχρονοι αλγόριθμοι και τεχνικές για τις περιπτώσεις αποκωδικοποίησης με ή χωρίς τη χρήση πληροφορίας αξιοπιστίας, για τα λαμβανόμενα δεδομένα στο δέκτη. Προτείνεται ένας αποδοτικός αλγόριθμος απαρίθμησης κόμβων, ο οποίος μπορεί να εγγυηθεί την αποκωδικοποίηση μέγιστης πιθανοφάνειας, για αποκωδικοποιητές σφαίρας με

ή χωρίς τη χρήση πληροφορίας αξιοπιστίας, για όλα τα πιθανά σενάρια λειτουργίας και συνθήκες καναλιού. Η προτεινόμενη λύση βασίζεται σε μια προκαθορισμένη σειρά επίσκεψης των κόμβων, σε μια και μόνο μονάδα υπολογισμού αποστάσεων, καθώς και σε κατάλληλα προσαρμοσμένη μετρική για το κλάδεμα του δέντρου. Η συγκεκριμένη τεχνική παρόλο που αυξάνει τον αριθμό των επισκέψιμων κόμβων, μειώνει την υπολογιστική πολυπλοκότητα ανά κόμβο και άρα τη συνολική πολυπλοκότητα της αποκωδικοποίησης. Επιπλέον, παρουσιάζεται η αρχιτεκτονική για το συγκεκριμένο αλγόριθμο απαρίθμησης κόμβων και συγκρίνεται η υλοποίησή του σε ASIC και συσκευές FPGA με υλοποιήσεις σύγχρονων τεχνικών απαρίθμησης, που μπορούν να εγγυηθούν την αποκωδικοποίηση μέγιστης πιθανοφάνειας. Η αποδοτική υλοποίηση οδηγεί σε μείωση του κόστους εφαρμογής και της κατανάλωσης ενέργειας με αποτέλεσμα τη πιθανή χρήση της και σε πιο πολύπλοκα συστήματα MIMO-OFDM .

ΘΕΜΑΤΙΚΗ ΠΕΡΙΟΧΗ: Αρχιτεκτονικές Πραγματικού Χρόνου για Επεξεργασία Δεδομένων Βασικής Ζώνης, Επεξεργασία Δεδομένων Βασικής Ζώνης για Συστήματα OFDM με χρήση Πολλαπλών Κεραιών

ΛΕΞΕΙΣ ΚΛΕΙΔΙΑ: Αρχιτεκτονικές Πραγματικού Χρόνου για Επεξεργασία Δεδομένων Βασικής Ζώνης για Συστήματα Πολλαπλών Κεραιών, Επεξεργαστής Ταχυ-μετασχηματισμού Fourier, Παράλληλες Αρχιτεκτονικές για Επεξεργαστές FFT, Αποκωδικοποίησης Σήματος σε Συστήματα Πολλαπλών Κεραιών, Αποκωδικοποιητής Σφαίρας

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In the last two decades, the increased demands for higher data-rates, better quality-of-service (QoS), higher network capacity and user coverage, lead the research on wireless communication systems, to advanced techniques for improving the spectral efficiency and link reliability. The most popular technology for advanced wireless systems is the use of multiple antennas at the transmitter and/or receiver side. The *Multiple-Input Multiple-Output (MIMO)* technology opens a new dimension (space), which if leveraged correctly can improve the wireless system performance substantially. Wireless communication industry has adopted MIMO technologies in several wireless standards, such as Wireless Local Area Network (WLAN IEEE-802.11), Worldwide interoperability for Microwave Access (WiMAX IEEE-802.16) and Long Term Evolution (LTE/LTE-Advanced). Furthermore, the MIMO technology is expected to play a key role in the upcoming IEEE 802.11ax standard and in the fifth generation (5G) mobile phone systems.

The majority of modern wireless communication systems use the *Orthogonal Frequency Division Multiplexing (OFDM)* technology to increase the spectral efficiency. In an OFDM system, the frequency domain bandwidth is divided into multiple non-overlapping sub-channels, each of which hosts a specific carrier (sub-carrier). Each of the transmitted data symbols is modulate a data sub-carrier, resulting in a frequency multiplexed signal, with each of the sub-channel spectra to be orthogonal to each other. This orthogonality of the sub-carriers ensures that the signals do not interfere with each other, when communicating over distortionless channels. Key advantages of the OFDM systems are the Time-Division Multiple Access (TDMA) [35], Frequency-Division Multiple Access (FDMA) [123], and

Code-Division Multiple Access (CDMA) [310], [92].

The MIMO technology can be used in OFDM wireless communication systems to increase capacity, coverage and reliability [23],[8]. The OFDM modulation converts a high-speed data channel into a number of parallel, lower-speed channels, and the processing required by MIMO techniques at higher speeds would be most manageable. The wide range of advantages provided by MIMO-OFDM systems comes at the expense of a substantial increase in the receiver complexity and sometimes in the transmitter complexity as well. The multiple processing paths of a MIMO-OFDM wireless architecture, along with advanced combined algorithms for high system performance over extreme channel conditions, increase the computational requirements and the implementation cost.

The rapid growth on the integrated circuit design has been contributing significantly to the trend of faster and more complex wireless communication systems during the past decades. The increased total number of transistors and thus logical functions per chip, not only enables the designers to integrate more complex algorithms, but also allows to speed up the circuits and to reduce the energy consumption. Both implementation cost and power efficiency steadily improved by the technology scaling, while large systems can be integrated in a single chip (*System-on-Chip (SoC)*). Furthermore, researchers provide algorithmic optimizations for complex MIMO processing functions, which reduce the computational requirements and the power consumption of the wireless communication system.

The increased transistor capacity on modern integrated circuits, provides the implementation flexibility of supporting multiple systems and protocols with a single chip and a *Software Defined Radio (SDR)* architecture. In SDR wireless systems part (or all) of the physical layer functionality can be implemented in software, running in one or more dedicated processors, which are implemented in a SoC. Computationally intensive and/or time-critical system functionality could be implemented as a hardware acceleration module, to reduce the processor utilization and the power consumption of the chip. These hardware units should be optimized to support the multiple protocol requirements, while having increased scalability, low implementation cost and low power consumption.

1.1 Motivation and Scope

The performance of the Fast Fourier Transform (FFT) and the Inverse FFT (IFFT) algorithms plays a key role in emerging wireless technology standards that are based on OFDM and MIMO-OFDM. The increased data-rates require executing the FFT/IFFT computations at relatively high speeds, while cost constraints imply the use of minimal resources. Moreover, low-latency calculation is crucial for several wireless systems (e.g. IEEE-802.11), while the significant high number of sub-carriers results in increased computational complexity for the FFT processor (e.g. DAB, DVB-T, DVB-T2).

The use of MIMO technology on OFDM-based wireless systems, increase the computational requirements of the system, for the FFT/IFFT processor, linearly with the number of spacial streams. A separate FFT computation should be performed for each of the data streams, by using as many FFT processors as the MIMO channels or by using less processors, running at higher speed and a time-sharing scheduling scheme. Several FFT architectures have been proposed in the literature for MIMO-OFDM systems, which use the idle clock cycles of the butterfly processors, to perform the computations for the additional data streams. These architectures reduce the total computational requirements of the FFT calculations, on a MIMO-OFDM system but they have low scalability in terms of MIMO streams and variable FFT lengths.

In the case of SDR systems, the FFT processor should support operational modes in which multiple protocols require FFT computations, sequentially or in parallel. A multi-protocol SDR system with several antennas can support several OFDM or MIMO-OFDM systems in parallel (e.g. 2x2 MIMO LTE and 3x3 MIMO IEEE-802.11ac). An efficient implementation requires a single FFT processor performing all the needed calculations for all the operational modes of the system. The advanced FFT architectures in the literature, which support multiple data streams (MIMO-OFDM systems), are unable to support efficiently multiple data streams from different protocols and variable FFT lengths (e.g. SDR mode with two MIMO streams of 2048-points FFT for protocol A and three MIMO streams of 128-points FFT for protocol B).

In a MIMO-OFDM communication system with M_T transmit and M_R receive antennas, each of the M_R antennas receives signal components of all M_T transmitting antennas. The

received signals include the line-of-sight signals as well as the multi-path reflected signal components causing destructive MIMO channel effects. In order to recover the original transmitted symbols, the MIMO receiver performs combined detection and synchronization, demodulation and decoding operations on the received signals from the M_R antennas. A core idea in MIMO-OFDM wireless communication systems, is to use the multiple antennas to transmit data in parallel, to increase the system data rate. The drawback to this system performance improvement, is the considerable increase in complexity of signal processing algorithms, needed for the separation of parallel data streams in the receiver side. MIMO detectors are used in the processing path of the MIMO receiver, in order to recover the transmitted symbols from the received symbol vectors.

The optimal MIMO detection, often is the most computational expensive algorithm in a MIMO-OFDM receiver's processing path, specially for systems with increased number of antennas and dense constellations. An efficient way to reduce the detection complexity, without compromising the system error-rate performance, is the use of the sphere decoding algorithm, which translates the detection problem to a constrained tree traversal search. For increased system error-rate performance, soft information, in the form of *Log Likelihood Ratios (LLRs)*, is used, instead of hard-decision bits for the MIMO detection and decoding processes.

A *Soft-Output* sphere decoder can be used in combination with a *Soft-Input* channel decoder (Viterbi, Turbo, LDPC, etc.) to increase the wireless system error-rate performance, for fading MIMO channels with high noise. The computational complexity of a *Soft-Output* sphere decoder is increased, in comparison to a hard-decision detection, due to the search of the *counter-hypothesis* tree nodes. Several sphere decoding algorithms have been proposed in the literature, which try to reduce the high complexity of the soft-decision MIMO detection process, by using alternative tree traversal strategies, efficient node enumeration techniques or other complexity reduction schemes.

Advanced sphere decoding algorithms are used in *Iterative* MIMO receiver architectures, in which iterations are performed between the MIMO detection and the channel decoder, resulting in increased system error-rate performance over extreme MIMO channels. The exchange of soft-information between the channel detector and decoder, leads to the use of *Soft-Input Soft-Output (SISO)* sphere decoding algorithms. In such cases the feed-

back from the channel decoder, in the form of LLRs, should be considered in the detection process of every iteration, resulting in increased computational complexity, in comparison to the soft output MIMO detectors.

A key processing step for a sphere decoder algorithm, is the tree node ordering or *node enumeration*, which has an impact on the decoder error-rate performance, throughput, complexity and power consumption. The Schnorr and Euchner Enumeration (SEE), in which the children nodes of a parent tree node, are visited in ascending order of their partial Euclidean distances, is one of the most efficient node ordering schemes, which can guarantee the optimal detection with reduced total number of visiting nodes. For sphere decoders with soft information, the enumeration process utilize most of the computational complexity and an efficient node ordering results in increase decoding throughput.

A few enumeration schemes, in the literature, reduce the enumeration complexity of the SEE by exploiting geometrical characteristics of the constellation map, without compromising the optimal solution. In most of the cases, the constellation points are splitted to subsets, and the evaluation process of the next enumerated node, considers only one candidate node from each subset. Based on the total number of subsets, the computational complexity can be reduced, in comparison to a *Full Enumeration and Sort (FES)* scheme. Several techniques have been proposed in the literature, in which the complexity reduction is based on calculations of non Euclidean distances or on pre-calculated visiting order of the constellation points. These schemes, reduce further the complexity, in comparison to the FES technique but they can not guarantee the *maximum-likelihood (ML)* or the *maximum a-posteriori (MAP)* detection performance, resulting in decreased error-rate performance, for specific channel conditions.

1.2 Key Contributions

This thesis tackles with the two of the most computational expensive processing units of a MIMO-OFDM system, the FFT/IFFT processor and the MIMO detector.

FFT/IFFT processor

Single FFT processor and time-sharing scheduling schemes have been used to handle the multiple data streams, of a MIMO-OFDM system. These architectures requires higher clock frequencies, for the FFT processor, proportional to the total number of MIMO channels. Multiple processors with more complex scheduling schemes, can reduce the extreme high clock frequencies, for large MIMO-OFDM systems, with increased hardware resources utilization and high power consumption. The efficient use of the idle clock-cycles of an FFT processor, for the computations of additional input data streams, increases the utilization of the butterfly processor, with limited scalability and usability in large MIMO-OFDM and SDR systems.

In this study, a detailed overview of the state of the art FFT architectures for multiple input data streams, is presented in Section 4. Each of the architectures is analyzed, in terms of computational complexity, processing latency, memory requirements, scalability and usability. An efficient and scalable in-place FFT architecture [252], [134], supporting multiple variable-length input data streams is proposed, in which a novel conflict-free addressing scheme is used to reduce the complexity of the FFT control unit, and to simplify the interconnection network between the multiple butterfly units and memory banks. The proposed FFT architecture can be tuned to support efficiently, large MIMO-OFDM or SDR systems, while a specific scheduling scheme can be implemented to optimize the processing latency, at run-time based on the total number of data streams and the FFT length of each stream. Furthermore, the proposed memory-based FFT architecture can support continuous flow operation for multiple data streams with different FFT lengths, with low memory requirements and parallel, normal-order output of the data streams, in comparison to the stream sequential output of the state of the art pipeline FFT architectures.

The reduced number of radix-2 butterfly processors and the efficient processor utilization for the FFT computations of multiple variable-length data streams, is a key element for an efficient implementation of an SDR MIMO-OFDM system. The increased scalability of the proposed architecture and the support of multiple variable-length FFT calculations, different for each of the data streams, is a unique characteristic, which make the proposed architecture easily adaptable to any multi-protocol and/or MIMO, SDR OFDM system implementation.

MIMO detector

The exponential computational complexity of the MIMO detection, based on exhaustive search, is prohibitive for real-time implementations, for large MIMO systems and/or dense constellations. The sphere decoding algorithm transforms the MIMO detection problem to a constrained tree search, resulting in reduced computational complexity for real-time implementations, without compromising the optimal solution, even in the cases of dense constellations. Soft information (LLRs) is used in the MIMO detection and channel decoding process, to increase the error-rate performance and the reliability of the system, under extreme channel conditions. The use of soft information increases the complexity of the sphere decoder, resulting in reduced throughput and increased power consumption.

To reduce the total number of visiting tree nodes and hence the total computational complexity of the MIMO detection process, the Schnorr and Euchner Enumeration (SEE) scheme have been proposed, in which the nodes are visited in increasing order of their partial Euclidean distances. The computational expensive Euclidean distance calculations, should be minimized to increase the decoding throughput and limit the power consumption of the sphere decoder implementation. Several enumeration schemes have been proposed in the literature, in which the reduction of the total distance calculations and visiting tree nodes, results in decreased error-rate performance, due to the non-optimal MIMO detection. The state of the art enumeration algorithms, which are not compromise the optimal solution, are computational expensive due to the required multiple distance calculation units, several comparison modules and/or complex memory structures (e.g. priority queues).

The proposed low-complexity, LUT-based enumeration technique [214] can guarantee the optimal solution (*ML* or *MAP*), with the use of only one distance calculation unit and small look-up tables. The specific algorithm results in increased number of visiting tree nodes, with a portion of the computational complexity per node, compared to other optimal enumeration schemes, and a reduced overall computational complexity for the MIMO detection process, for any scenario and channel condition. Furthermore, due to single distance calculation module the proposed implementation has reduced power consumption, compared to other enumeration schemes, particularly for small sphere constraint radius, which is the common case for sphere decoding algorithms with radius-update mechanism. Finally, the specific implementation can be reconfigured to compute near-optimal solution, by com-

promising the ML or MAP performance, to further reduce the power consumption of the MIMO detection unit, with specific channel conditions.

1.3 Organization of the Thesis

The remainder of this thesis is organized as follows. Chapter 2 is a brief introduction to MIMO wireless communication systems. In the first section a brief history of the wireless systems and the use of multiple antennas in the transmitter and/or receiver is presented. The next section is an introduction to the concept of the parallel transmission of data and the *Orthogonal Frequency Division Multiplexing (OFDM)* systems. The numerous advantages of the OFDM wireless systems are presented and the use of MIMO techniques for these systems is analyzed, in term of increased throughput, reliability and Quality-of-Service. In the last section of the chapter, the baseband processing of an OFDM wireless system is presented and the transmitter and receiver processing paths are analyzed. For the case of multiple transmit/receive antennas (MIMO-OFDM wireless systems) the complex multiple processing paths of the baseband are illustrated and the high-complexity modules are analyzed.

In Chapter 3 an introduction to the *Fast Fourier Transform (FFT)* algorithm is presented along with the most common FFT processor architectures. The first section of the chapter presents the different decompositions of the FFT algorithm along with the advantages and disadvantages, in terms of computational complexity. The most common FFT architectures are analyzed in the next section. The pipeline FFT architectures are presented in the first subsection, while the memory-based FFT architectures are analyzed in the second half of the section. Several FFT architectures are presented in this section with various computational complexity specifications, which can meet different system requirements. Advantages and disadvantages of each of the architectures are analyzed, in terms of throughput, processing latency and implementation cost.

The first section of Chapter 4 includes an overview of the common FFT architectures which are used in MIMO-OFDM systems. Several pipeline FFT processors can be used with multiple input streams, with the introduction of a time scheduling mechanism, or can be tailored for specific MIMO-OFDM systems by reordering the input data streams and ex-

ploiting the idle butterfly processing time. The next section presents an efficient and scalable memory-based FFT architecture for SDR/MIMO OFDM systems, with variable FFT length support. A novel conflict-free memory addressing scheme minimizes the processing latency, reduces the memory requirements of the FFT processor and results in an efficient and fully scalable architecture. The proposed FFT processor can be used in several multi-protocol (SDR) and/or multi-stream (MIMO) OFDM systems, with reduced implementation cost and increased computational efficiency. The next section of the chapter presents an analysis of the fixed-point performance of an FFT processor, and a detailed comparison of several techniques which are used to increase the SQNR performance of the FFT computations. In the final section of the Chapter 4, a detailed implementation comparison of the proposed FFT processor with state-of-the-art MIMO FFT architectures, is presented. Both FPGA and ASIC implementations are considered, while several architectures are compared in terms of implementation cost, performance efficiency and scalability.

Chapter 5 includes a brief introduction to the MIMO detection problem and the techniques used for the recovery of the multiple transmitted data streams on *Space Division Multiplexing (SDM)* MIMO wireless systems. The second section of the chapter analyzes the MIMO system model and presents the mathematical equations for the problem of the maximum likelihood MIMO detection. The next section provides an overview of the efficient *Sphere Decoding Algorithm (SDA)*, which transforms the MIMO detection problem to a tree search problem with reduced computational complexity, compared to the exhaustive search for the ML solution. Furthermore, the most common tree traversal strategies, for the implementation of the SDA are presented and compared in terms of complexity and decoding performance. The following section analyzes the use of soft information in the sphere decoding algorithms for increased decoding performance of the MIMO wireless system. *Soft-Output* sphere decoders can increase the reliability, of the wireless system, with the use of *Log-Likelihood Ratio (LLR)* values, instead of hard-decoded bits. The concept of the *Iterative Decoding* is presented with the use of *Soft-Input Soft-Output (SISO)* sphere decoding algorithms, which can deliver exact MAP performance even with extreme channel conditions. The final section of the Chapter 5 presents several complexity optimization techniques for various sphere decoding algorithms, which can reduce the implementation cost and the decoding latency, with or without a bit-error rate performance penalty.

Chapter 6 provides an overview of the most computational demanding process of the sphere decoding algorithm, the node enumeration. An efficient way to visit the tree nodes is in ascending order of their partial Euclidean distances (PED), to minimize the number of visiting nodes, avoid redundant computations and increase the decoding throughput. For high order, non-constant amplitude constellations, the enumeration process requires the calculation and sorting of all the partial Euclidean distances (*Full Enumeration and Sort (FES)*). State-of-the-art enumeration schemes, avoid FES by performing PED computations and sorting to a subset of the constellation points, which are selected by exploiting geometrical characteristics of the constellation map. These enumeration techniques are presented in the first four sections of Chapter 6, along with a comparison in terms of bit-error-rate decoding performance and implementation cost. In the fifth section of Chapter 6, an advanced LUT-based enumeration technique is presented, which reduce the total computational complexity of the sphere decoder without compromising the detection performance. This novel enumeration scheme can be used for Hard and Soft Output sphere decoders and it can be adapted for the case of iterative decoding (SISO). The next section provides an algorithmic performance evaluation for two versions of the proposed enumeration technique, one which can guarantee the ML/MAP solution ("exact") and one which further reduce the computational complexity of the sphere detection, with a small performance penalty ("approximate"). The final section of the chapter presents a comparison for FPGA and ASIC implementations, for the two version of the proposed enumeration scheme, in terms of complexity and power consumption.

Finally, Chapter 7 presents the conclusions obtained throughout this thesis and includes some guidelines for future research lines, in the area of the efficient baseband processing architectures for SDR and MIMO wireless communication systems.

Chapter 2

MIMO Wireless Communication Systems

The explosion of the communication market has irreversibly marked the end of the second millennium. Everyone now recognizes that we are in the beginning of an era of data communication. Wireless system designers are faced with a number of challenges. These include the limited availability of the radio frequency spectrum, a complex time-varying wireless environment (fading and multipath), the increased demand for higher data rates, better quality of service (QoS), higher network capacity and user coverage. Several innovative techniques are proposed to improve the spectral efficiency and link reliability. The use of multiple antennas at the receiver and/or transmitter in a wireless system, known as space-time or multiantenna communications is an emerging technology that promises significant improvements in these measures.

2.1 Wireless Communication Systems

Maxwell proposed, in 1861, a mathematical theory of electromagnetic waves. A practical demonstration of the existence of such waves was performed by Hertz in 1887, using stationary waves. The first radio telegraph was built and demonstrated by Marconi, in the summer of 1895. In the next few years, Marconi integrated many new technologies into his sophisticated radio equipment, including the diode valve developed by Fleming, the crystal

detector, continuous wave transmission developed by Poulsen,Ferssenden and Alexanderson, and the triode valve or audion developed by Forrest.

The installation of the first 2MHz land mobile radiotelephone system in 1921, by the Detroit Police Department for police car dispatch, was the first civilian use of wireless technology. In 1933 Armstrong invented the frequency modulation (FM), which made possible high quality radio communications. As demand for public wireless services began to grow, the Improved Mobile Telephone Service (IMTS) using FM technology was developed by AT&T. These were the first mobile systems to connect with the public telephone network using a fixed number of radio channels in a single geographic area. Extending such technology to a large number of users needed excessive bandwidth. A solution was found in the cellular concept (cellularization), conceived by Ring at Bell Laboratories in 1947. This concept required dividing the service area into smaller cells, and using a subset of the total available radio channels in each cell. The first high capacity analog cellular telephone system called the Advanced Mobile Phone Service (AMPS), was proposed by AT&T in 1970. Mobile cellular systems have evolved rapidly since then, incorporating digital communication technology and serve more than one billion subscribers worldwide.

The use of multiple antennas at the transmitter and /or receiver in a wireless communication link opens a new dimension − space, which if leveraged correctly can improve performance substantially. The use of multiple receive antennas for diversity goes back to Marconi and the early radio pioneers. So does the realization that steerable receive antenna arrays can be used to mitigate co-channel interference in radio systems. During and after the World War II, the use of antenna arrays was an active research area in radar systems. The arrival of digital signal processors in the 1970s, results in development of more sophisticated systems with adaptive signal processing at the wireless receiver for improving diversity and interference reduction, mainly for military applications. In 1994 Paulraj and Kailath proposed a technique for increasing the capacity of a wireless link using multiple antennas at both the transmitter and the receiver, while in 1996 Roy and Ottersten proposed the use of base-station antennas to support co-channel users. These ideas along with the fundamental research done at Bell Labs began a new revolution in information and communications theory in the mid 1990s. The goal is to approach performance limits and to explore efficient but pragmatic coding and modulation schemes for wireless links using multiple antennas.

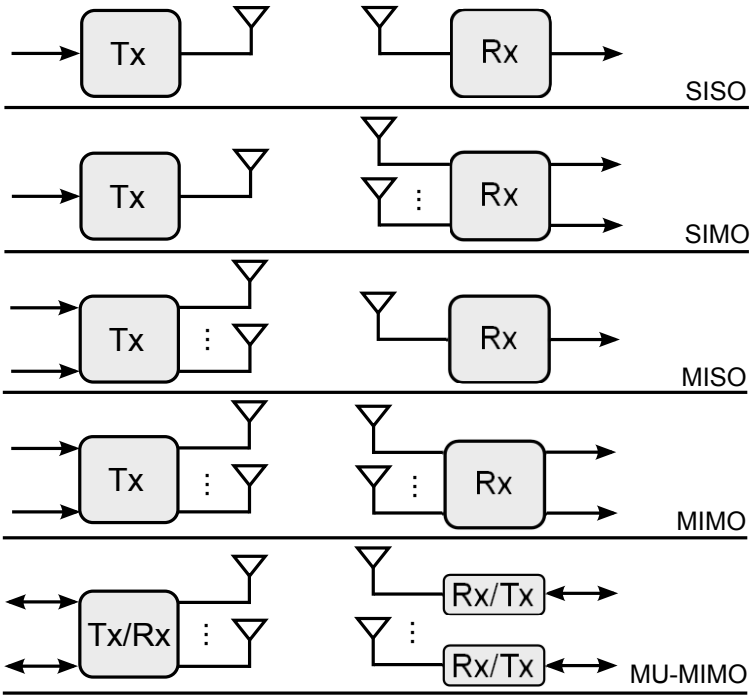


Figure 2.1 Antenna configurations in Space-Time wireless systems

Figure 2.1 shows the different antenna configurations for *Space-Time (ST)* wireless links. The *SISO* (Single Input Single Output) is the familiar wireless configuration in which only one antenna is used in both the transmitter and receiver. In *SIMO* (Single Input Multiple Output) wireless systems multiple antennas (M_R) are used only in the receiver, while in *MISO* (Multiple Input Single Output) systems multiple antennas (M_T) are used only in the transmitter. In a *MIMO* (Multiple Input Multiple Output) wireless system, multiple (M_T) transmit antennas and multiple (M_R) receive antennas are used. Finally, the MU-MIMO (Multi User MIMO) configuration refers to the case where a base-station with multiple (M) antennas communicates with P users each with one or more antennas.

In wireless systems with multiple antennas on the receiver, the transmitter or both, an average increase in the SNR at the receiver can be achieved with the coherent combining effect of multiple antennas. This signal gain is known as *Array gain*. In a *SIMO* system signals arriving at the receive antennas have different amplitudes and phases. The receiver can combine the signals coherently so that the resultant signal is enhanced. The average increase is signal power at the receiver is proportional to the number of receive antennas. In wireless systems with multiple transmit antennas (*MISO* or *MIMO*), array gain exploitation

requires channel knowledge at the transmitter.

Wireless channels can reduce significantly the signal power in random times. These channel fluctuations, also known as fades (fading channels), can be handled with *diversity*. Receive antenna diversity can be used in *SIMO* systems in which receiver see independently faded versions of the same signal. The receiver is able to combine these signals so that the resultant signal exhibits considerably reduced amplitude variability (fading), in comparison with the signal at any of the antennas. The *diversity order* is the number of independently fading branches, and is equal to the number of receive antennas in *SIMO* wireless systems. Transmit diversity is applicable to *MISO* systems with or without channel knowledge at the transmitter. Suitable design of the transmitted signal is required to extract diversity. *ST diversity coding* is a transmit diversity technique that relies on coding across space (transmit antennas) to extract diversity in the absence of channel knowledge at the transmitter. Utilization of diversity in *MIMO* wireless systems requires a combination of the receive and transmit diversity described above.

2.2 OFDM Introduction

The concept of parallel transmission of data over dispersive channels was first mentioned as early as 1957, in the pioneering work of Doelz et al. [70], while the first OFDM (Orthogonal Frequency Division Multiplexing) schemes date back to the 1960s, which were proposed by Chang [38] and Saltzberg [250]. The frequency domain bandwidth is divided into a number of non-overlapping sub-channels, each of which hosts a specific carrier, referred as subcarrier. Each subcarrier is modulated by a data symbol, resulting in a frequency-multiplexed signal, with each of the sub-channel spectra to be orthogonal to each other. This ensures that the subcarrier signals do not interfere with each other, when communicating over perfectly distortionless channels, as a consequence of their orthogonality.

The first OFDM schemes [38], [250], [37] had increased implementation complexity, due to the required banks of sinusoidal subcarrier generators and demodulators. In 1971, Weinstein and Ebert [324] suggested that the Discrete Fourier Transform (DFT) can be used for the OFDM modulation and demodulation processes, which significantly reduces the implementation complexity of OFDM. Peled and Ruiz [231], in the early 1980s, proposed a

simplified frequency domain data transmission method using a cyclic prefix-aided technique and exploited reduced-complexity algorithms for achieving a significantly lower computational complexity than that of classic single-carrier time-domain Quadrature Amplitude Modulation (QAM) modems. Hirosaki in [101] proposed a subchannel-based equalizer for an orthogonally multiplexed QAM system, while a DFT-based implementation of an OFDM system was introduced in [102], on the basis of which, a so-called group-band data modem was developed [103]. The performance of OFDM modems in mobile communication channels was investigated by Cimini [56] and Kalet [132], while Alard and Lassalle [5] introduce the OFDM concept in digital broadcasting systems, which was the pioneering work of the European DAB standard established in the mid-1990s.

Some key advantages of the OFDM over other widely used wireless access techniques, are Time-Division Multiple Access (TDMA) [35], Frequency-Division Multiple Access (FDMA) [123], and Code-Division Multiple Access (CDMA) [310], [92]. In an OFDM system the radio channel is divided into many narrowband, low-rate, frequency-non-selective subchannels or subcarriers [24], so that multiple symbols can be transmitted in parallel, while maintaining a high spectral efficiency. Information for different users can be delivered with the use of different subcarriers, resulting in a simple multiple-access scheme known as Orthogonal Frequency-Division Multiple Access (OFDMA) [138],[146],[34]. This enables different media such as video, graphics, speech, text or other data to be transmitted within the same radio link, depending on the specific types of services and their Quality-of-Service (QoS) requirements.

The implementation flexibility and the low complexity of the transmission and reception, as well as the high performance, render OFDM highly attractive candidate for high-data-rate communications over time-varying frequency-selective radio channels. In classic single-carrier systems, complex equalizers are used at the receiver, for the reduction of the Inter-Symbol Interference (ISI) introduced by multi-path propagation. By contrast, when using a cyclic prefix [231], OFDM exhibits a high resilience against the ISI. Incorporating channel coding techniques into OFDM systems, which results in Coded OFDM (COFDM) [195], [170], allows us to maintain robustness against frequency-selective fading channels, where busty errors are encountered at specific subcarriers in the frequency domain.

However, besides its significant advantages, OFDM also has a few disadvantages. One

problem is the associated increased Peak-to-Average Power Ratio (PAPR) in comparison with single-carrier systems [91], requiring a large linear range for the OFDM transmitter's output amplifier. In addition, OFDM is sensitive to carrier frequency offset, resulting in Inter-Carrier Interference (ICI) [205]. Several techniques has been proposed in the literature to efficient handle these problems in OFDM systems.

2.2.1 MIMO-OFDM

The main parameters, by which the quality of a wireless link can be described are the transmission rate, the transmission range and the transmission reliability. By reducing the transmission range and reliability we can increase the transmission rate. On the other hand, transmission range may be extended at the cost of a lower transmission rate and reliability, while the transmission reliability may be improved by reducing the transmission rate and range. Nevertheless, the use of MIMO technology in OFDM systems, may simultaneously improve the above-mentioned three parameters, of the wireless link quality [50].

MIMO techniques can be used in OFDM communication systems to increase capacity, coverage and reliability [23],[8]. The processing required by MIMO at higher speeds would be most manageable using OFDM modulation, because OFDM converts a high-speed data channel into a number of parallel, lower-speed channels. MIMO-OFDM is the foundation for most advanced wireless local area network (wireless LAN) and mobile broadband network standards because it achieves the greatest spectral efficiency and, therefore, delivers the highest capacity and data throughput [8],[274].

The research of Gregory Raleigh was the first in which MIMO technology was used in OFDM systems. He proved in [242], that with a proper type of MIMO system the multipath propagation could be exploited, resulting in increased capacity for the wireless link. Before Raleigh's work, researchers was trying to identify modulation and coding schemes that perform robustly over time-varing, dispersive, multipath channels. Raleigh published additional research on MIMO-OFDM under time-varying conditions [245], [243], MIMO-OFDM channel estimation [129], [244], synchronization techniques for MIMO-OFDM systems [129] and the performance of the first experimental MIMO-OFDM system [243].

In his PhD dissertation a comparison of three MIMO systems was performed in terms of performance and computational complexity. The MIMO-OFDM system was the only one in

which the computational complexity, of the receiver, was grown log-linearly as data rate was increased, in comparison with the quadratically increase of the other two MIMO systems (QAM and DSSS). In 1996 Raleigh founded the *Clarity Wireless* and developed the specifications that led to the IEEE 802.16 (WiMAX) and LTE standards, both of which support MIMO. The first MIMO-OFDM chipset (which later led to the IEEE 802.11n standard) was implemented by the *Airgo Networks*, which was founded in 2001 by Raleigh. The new IEEE 802.11ac (WiFi) standard includes the MIMO-OFDM, and is expected, MIMO-OFDM to play a major role, in the upcoming IEEE 802.11ax standard and in the fifth generation (5G) mobile phone systems.

2.2.2 Multi-User MIMO

Another benefit of the MIMO technology is to be able to deploy multiple antennas base stations to support multiple users with one or more antennas per user terminal. This technique refers as *Multi-User MIMO* or *MU-MIMO*. The exploitation of the spatial dimension (spatial signature), makes it possible to identify the individual users, even when they are in the same time/frequency/code domains, resulting in increased system capacity. Figure 2.2 shows a Multi-User MIMO system with one Base Station and multiple user terminals. Each user simultaneously communicates with the Base Station, which is equipped with an array of antennas. This technology is also known as *Space-Division Multiple Access (SDMA)* [304],[249],[140], [67],[305].

The SDMA technique can be considered as a specific branch of the family of MIMO systems, where the transmissions of the multiple-transmitter antennas cannot be coordinated, simply because they belong to different users. The major advantages of the SDMA technology can be summarized as:

- *Range extension*: The coverage area of high-integrity reception, with an antenna array, can be significantly larger than that of any single-antenna-aided system. For a given geographic area the number of required cells can be substantially reduced with the use of SDMA technology.

- *Multi-path mitigation*: The SDMA systems can benefit from the MIMO technology to efficiently mitigate the detrimental effects of multi-path propagations. Furthermore,

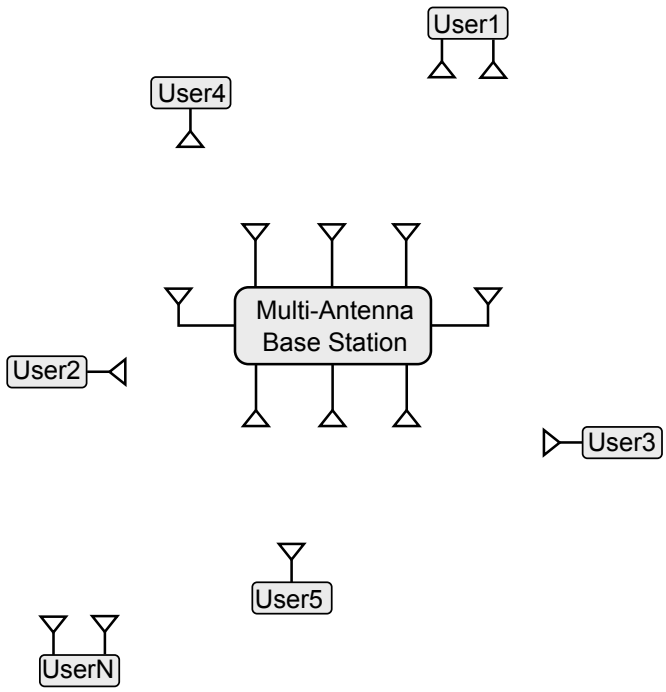


Figure 2.2 Multi-User MIMO (SDMA) System: One multi-antenna Base station with multiple user terminals with one or more antennas.

in specific scenarios the multi-path phenomenon can even be exploited to enhance the desired users' signals by employing efficient receiver diversity schemes.

- *Capacity increase*: Theoretically, SDMA can be incorporated into any existing multiple-access standard at the cost of a limited increase in system complexity, while attaining a substantial increase in capacity. For instance, by applying SDMA to a conventional TDMA system, two or more users can share the same time slots, resulting in a doubled or higher overall system capacity.

- *Interference suppression*: The interference imposed by other systems and by users in other cells can be significantly reduced by exploiting the desired user's unique, user-specific Channel Impulse Responses (CIRs).

- *Compatibility*: SDMA is compatible with most of the existing modulation schemes, carrier frequencies and other specifications. Furthermore, it can be readily implemented using various array geometries and antenna types.

The use of MU-MIMO or SDMA techniques on an OFDM system can be beneficial

due to the advantages of both technologies [6], [297], [306], [307], [7]. The new IEEE 802.11ac (WiFi) along with the LTE-Advanced (4G) standards use these new technologies to increase user data rates, network capacity and user coverage, while having better quality of service (QoS) for increased number of users. The OFDM and MU-MIMO techniques are in the core of the future standards as IEEE 802.11ax and fifth generation (5G) mobile phone system, and in the main focus of research for wireless communications [366], [62], [296], [64], [122], [364], [363], [180].

2.3 Baseband Processing

Orthogonal Frequency Division Multiplexing (OFDM) is a block modulation scheme, where a block of N information symbols is transmitted in parallel on N subcarriers. An OFDM modulator can be implemented by an Inverse Discrete Fourier Transform (I-DFT) on the block of information symbols followed by an Digital-to-Analog Converter (DAC). Typically, a cyclic prefix (CP) is introduced to the output of the IDFT, for the mitigation of the effects of the inter-symbol interference (ISI), caused by the channel time spread. For each transmission block a preceded cyclic prefix is introduced, containing a repetition of the last modulated subcarriers of the OFDM symbol. The length of the CP works as a guard interval between two consecutive OFDM symbols, while the repetition of the subcarriers in the CP allows the linear convolution of a frequency-selective multipath channel to be modeled as circular convolution, which results in low complexity frequency-domain processing at the receiver, for the tasks of channel estimation and equalization.

The physical layer of a telecommunication system includes a frequency-domain digital processing, a time-domain digital processing and the analog front-end. In a OFDM-based system, the Inverted Discrete Fourier Transform (IDFT) is used at the end of the frequency-domain processing, while the Digital-to-Analog Converter (DAC) is used at the start of the analog front-end (end of time-domain processing), at the transmission path. At the reception path the Analog-to-Digital Converter (ADC) is used at the end of the analog front-end, while the Discrete Fourier Transform (DFT) is used at the start of the frequency-domain processing (end of time-domain processing). Figure 2.3 shows a typical OFDM-based system transmission and reception processing paths. The frequency-domain digital processing

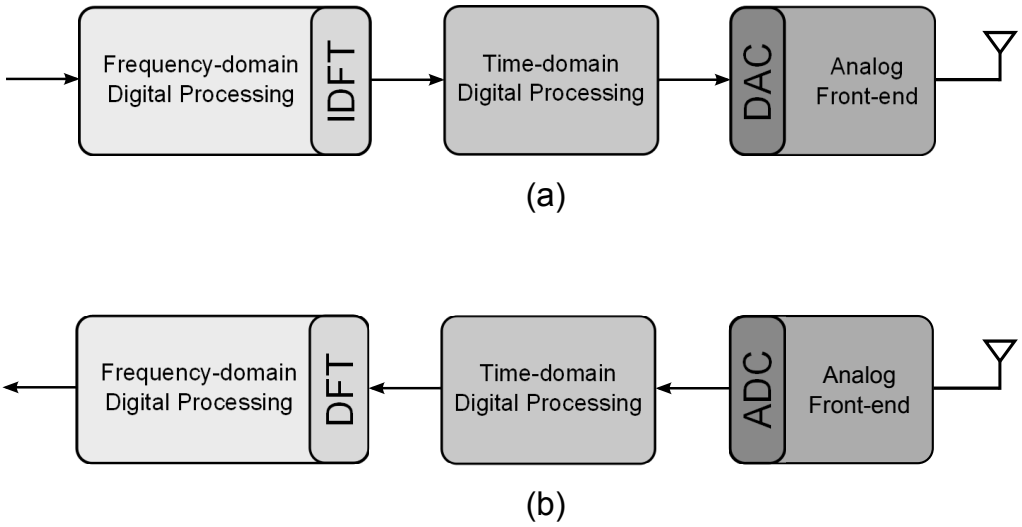


Figure 2.3 Example of a typical OFDM-based system Transmission (a) and Reception (b) processing paths

is also known as *Baseband Processing*.

The analog front-end path may contain analog amplifiers, synthesizers, mixers, filters, etc. along with the analog-to-digital and digital-to-analog converters. The time-domain processing path on the transmitter may contain FIR filters, IQ imbalances correction modules, predistortion modules, etc. while in the receiver path may contain FIR filters, Automatic-Gain Controllers (AGC), Synchronization and packet detection modules, Carrier-Frequency-Offset (CFO) estimation and correction units, Channel estimation and equalization units, Phase Noise estimators, etc.

2.3.1 OFDM Baseband processing

The Medium Access Control (*MAC*) layer forwards the data packet to be transmitted to the baseband processing path of the transmitter, in the form of a data bit stream. It is a common practice, for baseband processing, to "scramble" the bit stream of the data packet. This technique is used to eliminate consecutive zeros or ones in the bit stream, and in many cases it is implemented with a simple *Linear-Feedback Shift Register (LFSR)* unit. The output of the scrambler unit is forwarded to the *Forward-Error-Correction (FEC)* unit. This module contains the implementation of one or more error codes and constructs a new data bit stream, which contains information from the scrambled bit stream and additional embedded

information for the error correction procedure on the receiver processing path. The FEC module introduces additional information to the data bit stream and the selected coding rate indicates the amount of these additional information compared to the initial data bit stream. For specific error codes, which are sensitive to localized errors in the bit stream (like BCC, TurboCodes, etc), an additional module is used to shuffle the bit stream, based on specific permutations (*Interleaver*). The use of the interleaving technique eliminates the possibility, localized channel interference to produce localized data errors in the bit stream. For error codes which are robust to localized errors (like LDPC codes) this technique is not used.

The output data of the FEC module are forwarded to the constellation mapping unit, which maps block of bits to vectors in the IQ space (complex numbers), based on the selected constellation. State information about the channel (signal-to-noise ratio, interference, fading, etc.) are gathered by the MAC layer, based on several measurements on the transmitter (packet-error-rate, receiver channel feedback, etc.). Based on these information a specific constellation is selected for the next packet transmission. There are constellations which are robust to channel noise, interference and fading (*BPSK*,*QPSK*,*PSK*,*DPSK*) and constellations which are sensitive to low singal-to-noise ratios (SNR) and fading (*16-QAM*,*64-QAM*,*256-QAM*, etc.). A constellation map contains specific points in the IQ space, which are represented by IQ vectors. The indices of the constellation points are used for the mapping of the data bit stream. The more the points of a constellation map, the more bits can be mapped to a specific IQ vector (increased transmission throughput), with the disadvantage of increased sensitivity to channel noise. The IQ space has specific dimensions and more constellation points results in decreased distances between the points, and therefore increased possibility for demapping errors in the receiver, with high channel interference.

The IQ vectors at the output of the constellation mapping module are forwarded to the IDFT unit. The majority of the OFDM-based systems use a power of two OFDM symbol size for low complexity implementation of the IDFT module, by using the Inversed Fast Fourier Transform (*IFFT*). Each of the OFDM symbols, at the input of the IFFT unit, contains the data subcarriers (blocks of data bits mapped to IQ vectors), the null subcarriers (zero-value IQ vectors) and the pilot subcarriers (specific IQ vectors, which are known to the receiver). Null subcarriers at the “edges” of the OFDM symbols act as guard intervals between consecutive symbols, eliminating the Inter-Symbol-Interference, while null sub-

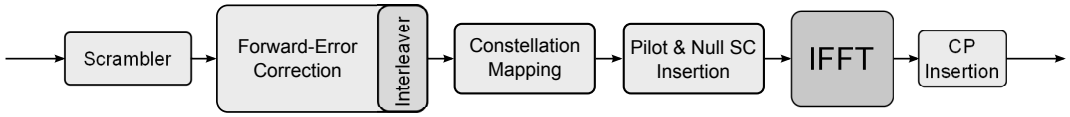


Figure 2.4 Example of an OFDM transmission baseband processing path

carriers at the "center" of the OFDM symbol decrease the implementation complexity of the Digital-to-Analog and Analog-to-Digital converters [219]. Pilot subcarriers are specific IQ vectors which are spread inside the OFDM symbols. The values of the pilot IQ vectors are known to the receiver and they can be constant for each symbol or can be produced based on a specific sequence, by using an LFSR unit. These known values are used in the receiver processing path for channel estimation and tracking, synchronization, carrier frequency offset (*CFO*), sampling clock offset (*SCO*) and Phase noise (*PHN*) estimation [60], [168], [178], [357], [239], [360], [206]. The output of the IFFT processor is a sequence of time-domain IQ samples. A portion of the start of the sequence, for each of the transmission symbols, is appended at the end of the sequence as Cyclic Prefix to eliminate the Inter-Symbol-Interference and reduce the complexity of the channel estimation and synchronization on the receiver processing path. Figure 2.4 shows an example of an OFDM transmission baseband processing path. The module with the higher computational complexity, in the transmitter processing path, is the IFFT unit, where for specific channel codes the FEC block can have increased complexity.

At the receiver, the time-domain processing path forwards the received IQ samples sequence to the baseband processing path. The cyclic prefix can be used for the synchronization of the symbols [258], while the time-domain IQ samples are forwarded to the FFT module. Carrier frequency offset, sample clock offset, phase noise and channel noise estimations can be calculated by using time-domain IQ samples or frequency-domain IQ vectors, depending on the selected algorithms [303], [33],[205], [42], [63], [215], [79]. The FFT unit outputs the IQ vectors, which can be equalized based on the above mentioned estimations, before the constellation demapper module. Furthermore, the pilot subcarriers can be used for *CFO* and *SCO* estimation and compensation and for channel tracking and channel estimation corrections, when the channel interference changes during a packet reception [188], [347].

In the case of rapidly time varying channels (mobile or vehicular systems) enhanced

techniques for channel tracking is used. Advanced algorithms for channel estimation and tracking, based on preamble symbols and pilot subcarriers, or the introduction of more complex pilot subcarriers in the OFDM symbols shown promising results in terms of system performance, with increased computational complexity [276], [253], [84]. The introduction of specific pilot OFDM symbols, in the middle of the packet transmission, or the channel tracking based on received data subcarriers (along with the pilot subcarriers) increase the system reliability on mobile and vehicular channels with the drawback of decreased throughput and increased complexity [136], [71], [83]. An enhanced technique for increasing the performance and reliability of the system, in rapidly time varying wireless channels is the Decision-Feedback Equalization method. The receiver performs part of the transmission processing path, to the output data of the FEC decoder and calculates a possible channel interference for the specific OFDM symbol. Using this technique, the channel state information on the receiver, can be updated with the latest channel interference, providing a reliable channel tracking mechanism [53], [58], [165], [1], [26].

Based on the FEC decoder used, the constellation demapper can produce decoded bit stream (*Hard-Decision Demapper*) or stream of soft-information bits (*Soft-Decision Demapper*). The most common soft-information used in OFDM systems, is the *Log-Likelihood Ratio (LLR)* which indicates the probability for a specific bit to be "1" or "0". The FEC decoder unit uses this soft-information to calculate the probability of an error on a specific bit or block of bits, for more efficient error correction. Soft-Decision demappers and Soft-information FEC decoders have increased computational complexity, compared to Hard-Decision processing paths with the benefit of increased error correction capability. Based on the expecting channel conditions of a wireless system and complexity requirements, the soft-information processing paths are commonly selected to increase the system throughput, due to enhanced error correction capability with increased channel interference [299], [232], [212], [320], [248]. For specific error correction codes (BCC, Turbo, etc) a De-Interleaver module is used to un-shuffle the data bits or LLRs based on the same permutations which are used in the Interleaver module of the transmission processing path.

A more advanced Demapping-Decoding scheme is that in which the FEC decoder unit outputs also soft-information bits, which then forwarded to the Soft-Decision demapper for re-evaluation of the demapped values. The demapper module outputs new soft-information

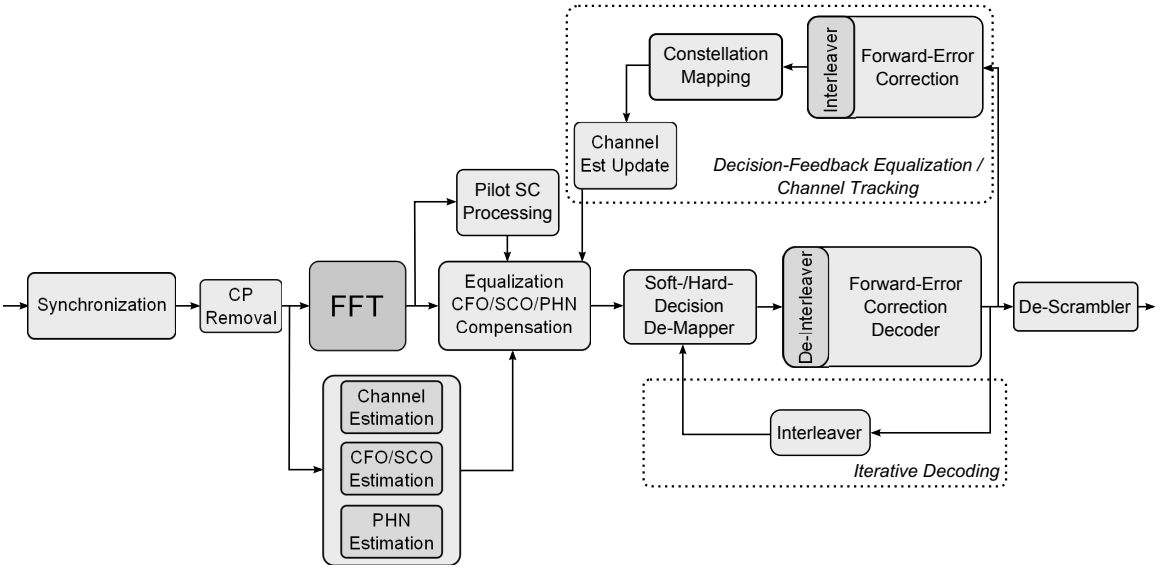


Figure 2.5 An example of an OFDM reception baseband processing path with channel, CFO, SCO and PHN estimation and compensation units, Pilot SC processing unit, Iterative Decoding mechanism and Decision-Feedback Equalization method for channel tracking.

bits with decreased probability of errors and the FEC decoder can evaluate the new soft-information for a more reliable and error-free output. This iterative process (*Iterative Decoding*) between the demapper and the decoder continue until the new soft-information, from the demapper, does not affect the output of the decoder. The iterative decoding techniques increase the system reliability in extreme channel conditions with the disadvantage of increased system complexity and latency [150],[295],[358], [362]. For error correction codes which use interleaving techniques the De-Interleaver/Interleaver modules are used in each iteration of the algorithm, for the un-shuffling/shuffling of the soft-information LLRs.

Figure 2.5 shows an example of an OFDM-based receiver baseband processing path. The *Iterative Decoding* and *Decision-Feedback Equalization* techniques are illustrated. The computation complexity is higher than that of the transmission processing path, with the FEC decoder, FFT and Channel Estimation units having increased complexity compared to other modules. Based on the selected algorithms the *CFO*/*SCO*/*PHN* estimation and compensation units and the Soft-Decision Demapper could have increased computational complexity.

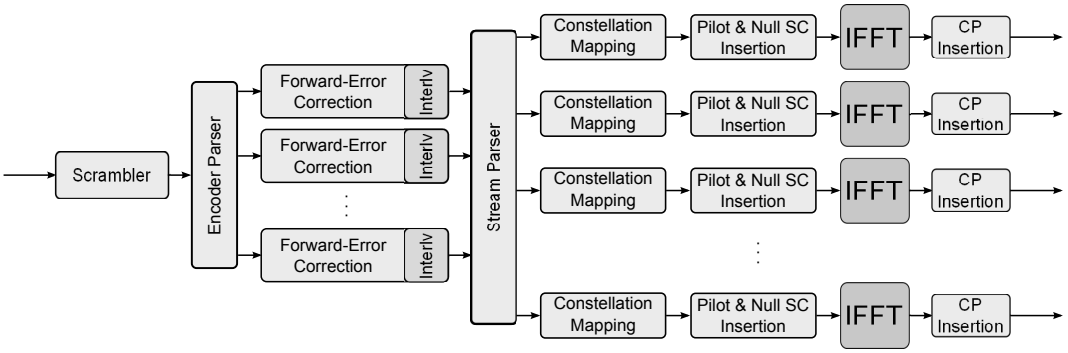


Figure 2.6 An example of a MIMO-OFDM transmission baseband processing path (Spatial Multiplexing MIMO mode)

2.3.2 MIMO-OFDM Baseband processing

In the case of MIMO-OFDM systems, multiple processing paths are commonly used for the processing of multiple data streams. Depending on the MIMO mode, the transmission processing path can be splitted in multiple paths either before or after the IFFT module. In MIMO cases with increased system throughput requirements (*spatial multiplexing* - *SM*), the data stream after the scrambler unit, is divided into several *Encoder Streams* , using multiple encoder units. For reducing the computational complexity on the receiver processing path, it is common to use less encoder streams than spatial MIMO streams (reducing the number of FEC decoder units). An example is the IEEE 802.11ac protocol, in which for 4 spatial streams a maximum number of 3 encoder streams are used. An additional unit (stream parser) is required for the permutation of the X encoder data streams to Y spatial data streams, while the total number of the encoder streams should have the same throughput as the total number of spatial streams. Figure 2.6 shows an example of a MIMO-OFDM transmitter baseband processing path, for the case of spatial multiplexing MIMO mode. In MIMO cases with increased system reliability requirements, the same information is transmitted over multiple antennas and the single processing path is commonly splitted to multiple ones, after the IFFT module.

In the receiver processing path more complex architectures are required, for a MIMO-OFDM system. In the common case the synchronization, channel estimation, *CFO*/*SCO*/*PHN* estimation and channel equalization can not be splitted easily to multiple units, due to the interference between the spatial streams (multistream interference - *MSI*) [8], [274], [141],

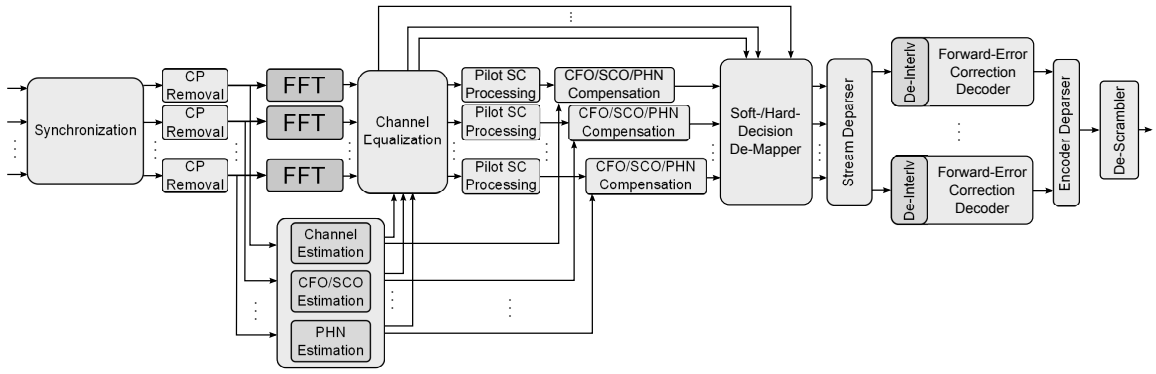


Figure 2.7 An example of a MIMO-OFDM receiver baseband processing path (Spatial Multiplexing MIMO mode)

[135], [164]. Advanced algorithms are used for these processing tasks with increased computational complexity [272], [284], [350], [204], [257], while more complex feedback architectures are used for channel estimation and tracking in rapidly time-varying channel conditions [20], [277], [279], [104].

The FFT calculation unit used in OFDM-based systems can be used also in MIMO-OFDM systems, for the computation of the transform in multiple streams. MIMO requirements demand FFT modules that can handle the increased throughput of the multiple spacial streams. Receiver architectures with multiple FFT processors, a single time-sharing FFT module or efficient data scheduling FFT schemes has been proposed, to handle the high throughput requirements of a MIMO-OFDM system. Moreover, the demapper unit from the OFDM baseband processing path, can also be used in MIMO-OFDM systems with increased throughput requirements. More advanced algorithms for *MIMO Detection* can be used to increase both system performance and reliability in several channel conditions [147], [77]. These algorithms consider the interference between the spacial streams and the estimated channel conditions and they estimate the most likely transmitted symbols [8]. The high throughput requirements of MIMO-OFDM systems in comparison with the increased computational complexity of these algorithms, is the main subject in several research articles [29], [88], [68], [19], [275], [105].

Figure 2.7 shows a typical example of a MIMO-OFDM receiver baseband processing path with multiple spatial and encoder streams. In several MIMO-OFDM systems the receiver architecture could be more complex, with advanced channel estimation and tracking

implementations, increased complexity CFO/SCO/PHN estimation and compensation units, iterative decoding schemes, etc.

Chapter 3

Fast Fourier Transform

The Fourier transform is widely used in physics, engineering, statistics and applied mathematics. A key property of the Fourier transform is its ability to allow one to examine a function or waveform from the perspective of both the time and frequency domains. The fast Fourier transform (FFT) is the efficient algorithm for the calculation of the discrete Fourier transform (DFT) of a sequence of N numbers. In digital signal processing the DFT is the most widely used transform and it has been applied in a wide range of fields such as noise reduction, global motion estimation, speech synthesis and recognition, music synthesis, cardiac patients diagnosis, digital audio/video broadcasting (DAB/DVB), orthogonal frequency division multiplexing (OFDM) systems.

J.W. Cooley and J.W. Tukey in 1965 [117] showed that the computational complexity of a N-point DFT can be reduced from $\mathcal{O}(N^2)$ to $\mathcal{O}(N \cdot \log N)$ by using the new FFT algorithm. This publication caught the attention of the science and engineering community but was not the first one to propose the FFT algorithm. In 1805, two years before the work of Jean Baptiste Joseph Fourier on representations of functions as infinite harmonic series, Carl Friedrich Gauss propose the techniques, that we now call the fast Fourier transform (FFT), for calculating the coefficients in a trigonometric expansion of an asteroid's orbit [98]. This algorithm is as general and powerful as the Cooley-Tukey common-factor algorithm and is, in fact, equivalent to a decimation-in-frequency algorithm adapted to a real data sequence. Nevertheless, the FFT algorithm had a revolutionary effect on many digital processing methods, and remains the most widely used method of computing Fourier transforms.

3.1 Introduction

The discrete Fourier transform (DFT) of a sequence $x[n]$ is calculated based on the following equation:

$$X[k] = \sum_{n=0}^{N-1} x[n] \cdot W_N^{nk}, \quad k = 0,1,\ldots,N-1, \quad where \quad W_N^{nk} = e^{-j(2\pi/N)kn} \tag{3.1}$$

The computational complexity of the DFT calculation can be reduced by decomposing the computation into smaller DFT calculations. By exploiting the symmetry and the periodicity of the twiddle factors (W_N^{kn}) we can reduce the required calculations. Algorithms in which the decomposition is based on decomposing the sequence $x[n]$ into successively smaller subsequences are called *decimation-in-time* algorithms, while the *decimation-in-frequency* algorithms are based on the decomposition of the sequence $X[k]$ into successively smaller subsequences [194].

Radix-2 Decimation-in-Time Decomposition

Assuming that N is an integer power of 2, i.e., $N = 2^v$ we can use the *decimation-in-time* algorithm to calculate the $X[k]$, by separating $x[n]$ into two $(N/2)$-point sequences consisting of the even-numbered points in $x[n]$ and the odd-numbered points in $x[n]$. From (3.1) we have:

$$X[k] = \sum_{n \ even} x[n] \cdot W_N^{nk} + \sum_{n \ odd} x[n] \cdot W_N^{nk} \tag{3.2}$$

With the substitution of variables $n = 2r$ for n even and $n = 2r+1$ for n odd,

$$\begin{aligned} X[k] &= \sum_{r=0}^{(N/2)-1} x[2r] \cdot W_N^{2rk} + \sum_{r=0}^{(N/2)-1} x[2r+1] \cdot W_N^{(2r+1)k} \\ &= \sum_{r=0}^{(N/2)-1} x[2r] \cdot \left(W_N^2\right)^{rk} + W_N^k \cdot \sum_{r=0}^{(N/2)-1} x[2r+1] \cdot \left(W_N^2\right)^{rk} \end{aligned} \tag{3.3}$$

We can substitute the W_N^2 with $W_{N/2}$, since:

$$W_N^2 = e^{-2j(2\pi/N)} = e^{-j2\pi/(N/2)} = W_{N/2} \tag{3.4}$$

From (3.3) and (3.4) we can evaluate the $X[k]$ as:

$$\begin{aligned} X[k] &= \sum_{r=0}^{(N/2)-1} x[2r] \cdot W_{N/2}^{rk} + W_N^k \cdot \sum_{r=0}^{(N/2)-1} x[2r+1] \cdot W_{N/2}^{rk} \\ &= G[k] + W_N^k \cdot H[k], \quad k = 0,1,\ldots,N-1 \end{aligned} \tag{3.5}$$

In (3.5) each of the sums is an $(N/2)$-point DFT, the first sum is the $(N/2)$-point DFT of the even-numbered points of the original sequence and the second sum is the $(N/2)$-point DFT of the odd-numbered points of the original sequence. Although the index k ranges over N values ($k = 0,1,\ldots,N-1$), each of the sums in (3.5), must be evaluated only for k between 0 and $(N/2)-1$, since $G[k]$ and $H[k]$ are each periodic in k with period $N/2$. After the calculation of the two $(N/2)$-point DFT ($G[k]$ and $H[k]$), we can combine them and compute the N-point DFT ($X[k]$). Figure 3.1 shows the decimation-in-time decomposition, based on (3.5), for an 8-point DFT, where the $G[k]$ designating the 4-point DFT of the even-numbered points and $H[k]$ designating the 4-point DFT of the odd-numbered points.

Based on (3.1) the direct computation of the DFT ($X[k]$) requires a total of N^2 complex multiplications and N^2 complex additions[1]. By comparison, (3.5) requires the computation of two $(N/2)$-point DFTs, which require $2(N/2)^2$ complex multiplications and approximately $2(N/2)^2$ complex additions, if we calculate the $(N/2)$-point DFTs with the direct method (eq. (3.1)). Extra computations are needed to calculate the final N-point DFT, which require N complex multiplications and N complex additions. The computation of (3.5) requires at most $N + 2(N/2)^2$ or $N + N^2/2$ complex multiplications and complex additions, which are less than the N^2 required by the direct computation (for $N > 2$).

If $N/2$ is even, in (3.5), we can use the same decomposition for the calculation of each of the $(N/2)$-point DFTs. The computation of $G[k]$ (and $H[k]$) can be done by first calculate two $(N/4)$-point DFTs and then combine the results. Thus, $G[k]$ in (3.5) would be represented as:

$$G[k] = \sum_{r=0}^{(N/2)-1} g[r] \cdot W_{N/2}^{rk} = \sum_{l=0}^{(N/4)-1} g[2l] \cdot W_{N/2}^{2lk} + \sum_{l=0}^{(N/4)-1} g[2l+1] \cdot W_{N/2}^{(2l+1)k}$$

[1] The total required complex additions are $N(N-1)$ but for large values of N, $(N-1)$ can be approximated by N.

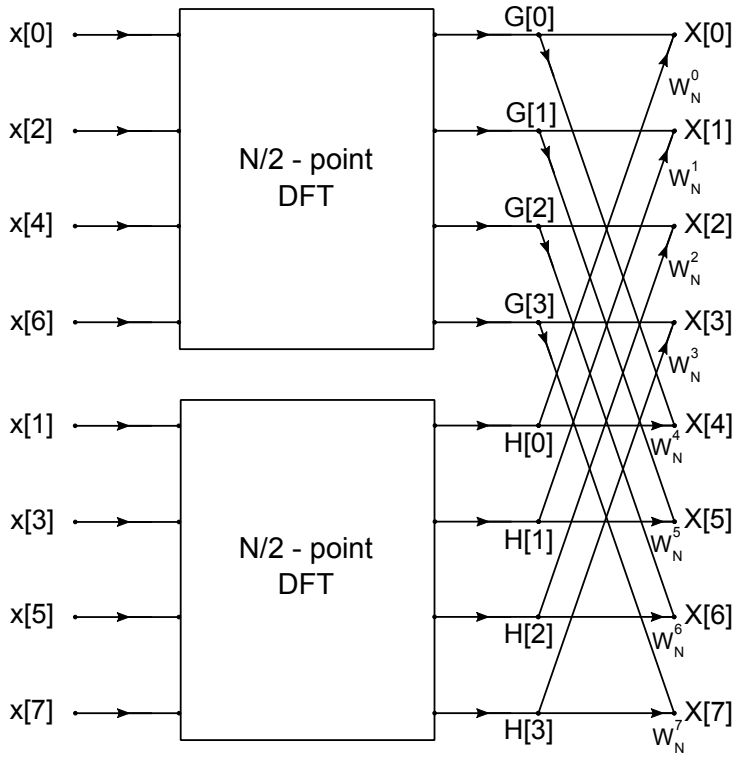


Figure 3.1 Flow graph of the decimation-in-time decomposition of an N-point DFT computation into two $(N/2)$-point DFT computations for N=8.

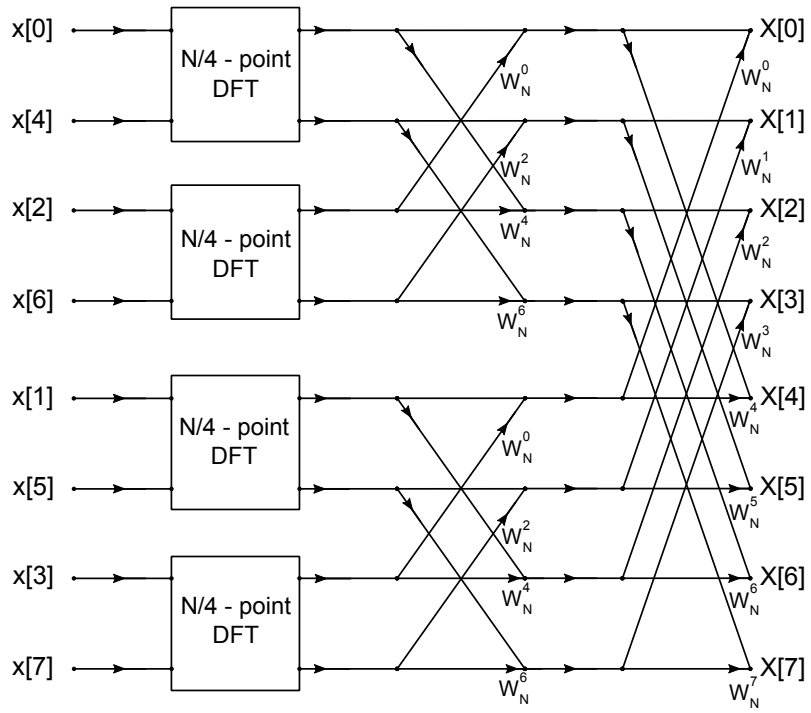


Figure 3.2 Flow graph of the decimation-in-time decomposition of an N-point DFT computation into four $(N/4)$-point DFT computations for N=8.

or

$$G[k] = \sum_{l=0}^{(N/4)-1} g[2l] \cdot W_{N/4}^{lk} + W_{N/2}^{k} \sum_{l=0}^{(N/4)-1} g[2l+1] \cdot W_{N/4}^{lk} \tag{3.6}$$

Similarly, $H[k]$ would be represented as:

$$H[k] = \sum_{l=0}^{(N/4)-1} h[2l] \cdot W_{N/4}^{lk} + W_{N/2}^{k} \sum_{l=0}^{(N/4)-1} h[2l+1] \cdot W_{N/4}^{lk} \tag{3.7}$$

Figure 3.2 shows the decimation-in-time decomposition, based on (3.6) and (3.7), for an 8-point DFT.

In the general case that N is an integer power of 2, we can decompose the DFT computation further until we were left with only 2-point transforms. This requires $v = log_2 N$ stages of computation. Figure 3.3 shows the complete decimation-in-time decomposition of an 8-point DFT computation. There are 3 stages of decomposition ($v = log_2 N = 3$) and each stage has N complex multiplications and N complex additions. Since there are $v = log_2 N$ stages, we have a total of $N \cdot log_2 N$ complex multiplications and additions.

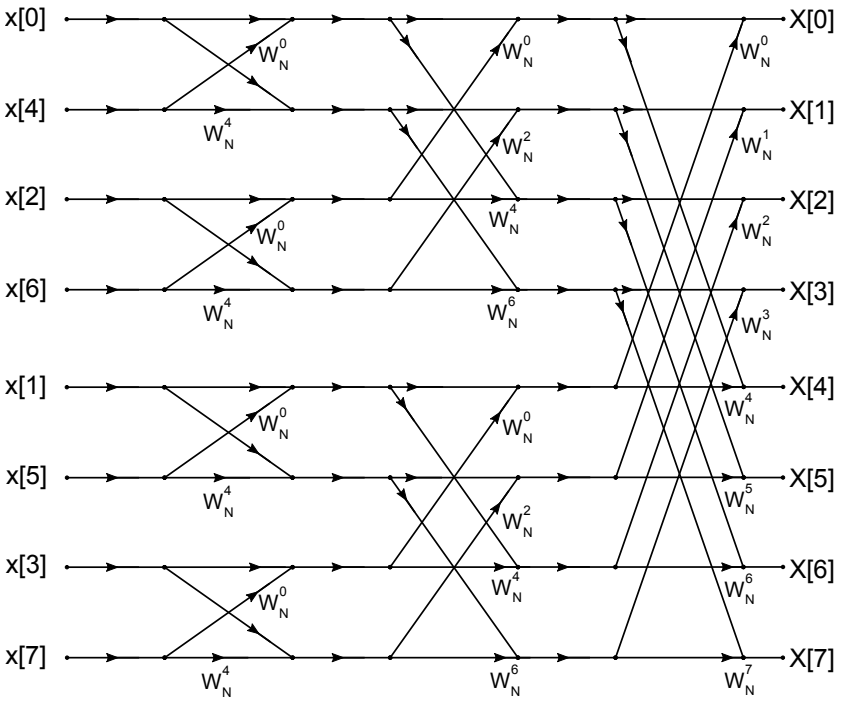


Figure 3.3 Flow graph of complete decimation-in-time decomposition of an 8-point DFT computation.

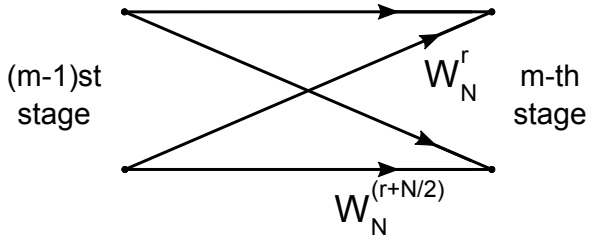


Figure 3.4 Flow graph of basic butterfly computation in Fig. 3.3

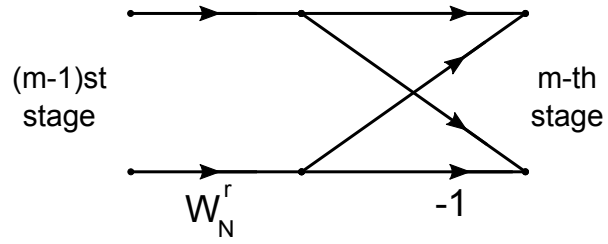


Figure 3.5 Flow graph of simplified butterfly computation requiring only one complex multiplication.

In proceeding from one stage to the next in Fig. 3.3, the basic computation is in the form of Fig. 3.4. A pair of values in one stage (m-th stage) is obtained by a pair of values in the preceding stage ((m-1)-th stage), where the coefficients are always power of W_N and the exponents are separated by $N/2$. The flow graph of Fig. 3.4 is an elementary computation and it is called *butterfly*. A more accurate name for this flow graph is *Decimation-in-Time Radix-2 butterfly*, due to the fact that it is a computation with 2 inputs and 2 outputs and it is based on the decimation-in-time decomposition.

Since

$$W_N^{N/2} = e^{-j(2\pi/N)N/2} = e^{-j\pi} = -1 \tag{3.8}$$

the factor $W_N^{r+N/2}$ can be written as

$$W_N^{r+N/2} = W_N^{N/2} \cdot W_N^r = -W_N^r \tag{3.9}$$

This will result in a simplified butterfly computation which requires only one complex multiplication instead of two. The simplified butterfly is shown in Fig. 3.5. By using the basic flow graph of Fig. 3.5 in the computation of a N-point DFT (Fig. 3.3) we can reduce the number of required complex multiplications by a factor of 2. Figure 3.6 shows the flow graph of an 8-point DFT computation using the simplified butterfly computations. Note that there are trivial complex multiplications in the flow graph of Fig. 3.6. The multiplication with $W_N^0 = 1$ can be removed while the multiplication with $W_N^{N/4} = -j$ can be evaluated by exchanging the real and imaginary parts of the number.

Figure 3.6 shows that for the calculation of a N-point DFT we need $N/2$ coefficient values (W_N^r). These values are independent from the input values of the DFT and can be precomputed and stored in a look-up-table (LUT). Several DFT/FFT implementations use

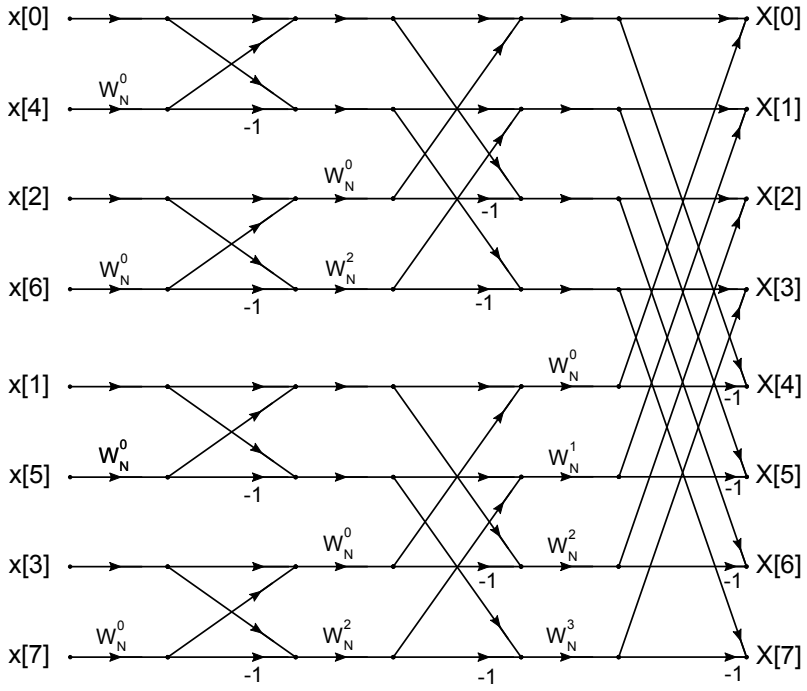


Figure 3.6 Flow graph of 8-point DFT using the simplified butterfly computation of Fig. 3.5.

a LUT structure to store the coefficient values to speedup the computation. Nevertheless, there are cases that the LUT structure is too big (for large values of N) or the memory requirements prohibit the use of precomputed coefficient values. We can exploit the symmetry and periodicity of the coefficients to reduce the required number of precomputed coefficient (twiddle factors) values. The twiddle factors for the computation of a N-point DFT are:

$$W_N^r = e^{\frac{-j2\pi r}{N}} = \cos\frac{2\pi r}{N} - j\sin\frac{2\pi r}{N}\ ,\ 0 \le r \le \frac{N}{2} - 1 \tag{3.10}$$

We can split the coefficients into two groups and calculate their values based on $N/4$ precomputed sine and cosine values:

$$\begin{aligned} W_N^r &= \cos\tfrac{2\pi r}{N} - j\sin\tfrac{2\pi r}{N} \\ W_N^{r+N/4} &= \sin\tfrac{2\pi r}{N} - j\cos\tfrac{2\pi r}{N} \end{aligned} \quad , 0 \le r \le \tfrac{N}{4} - 1 \tag{3.11}$$

The required precomputed sine and cosine values are $N/4 + N/4 = N/2$ real values compared to $N/2$ complex values (or N real values) of (3.10).

Furthermore, we can reduce the required pre-calculated values by splitting the twiddle factors in four groups:

$$\begin{aligned} W_N^r &= \cos\tfrac{2\pi r}{N} - j\sin\tfrac{2\pi r}{N} \\ W_N^{r+N/8} &= \tfrac{\sqrt{2}}{2}\left(\cos\tfrac{2\pi r}{N} + \sin\tfrac{2\pi r}{N}\right) + j\tfrac{\sqrt{2}}{2}\left(\sin\tfrac{2\pi r}{N} - \cos\tfrac{2\pi r}{N}\right) \\ W_N^{r+N/4} &= \sin\tfrac{2\pi r}{N} - j\cos\tfrac{2\pi r}{N} \\ W_N^{r+3N/8} &= \tfrac{\sqrt{2}}{2}\left(\sin\tfrac{2\pi r}{N} - \cos\tfrac{2\pi r}{N}\right) - j\tfrac{\sqrt{2}}{2}\left(\cos\tfrac{2\pi r}{N} + \sin\tfrac{2\pi r}{N}\right) \end{aligned} \tag{3.12}$$

with , $0 \le r \le \frac{N}{8} - 1$, or

$$\begin{aligned} W_N^r &= \cos\tfrac{2\pi r}{N} - j\sin\tfrac{2\pi r}{N} \\ W_N^{r+N/8} &= A + jB \\ W_N^{r+N/4} &= \sin\tfrac{2\pi r}{N} - j\cos\tfrac{2\pi r}{N} \qquad , \ 0 \le r \le \tfrac{N}{8} - 1 \\ W_N^{r+3N/8} &= B - jA \end{aligned} \tag{3.13}$$

where $A = \frac{\sqrt{2}}{2}\left(\cos\frac{2\pi r}{N} + \sin\frac{2\pi r}{N}\right)$ and $B = \frac{\sqrt{2}}{2}\left(\sin\frac{2\pi r}{N} - \cos\frac{2\pi r}{N}\right)$.

The required precomputed values for this case are the $N/8 + N/8 = N/4$ sine and cosine real values and the constant value of $\frac{\sqrt{2}}{2}$, compared to the $N/2$ real values of (3.11) and N real values of (3.10). The extra computational steps of (3.13) and (3.11) increase the complexity of the FFT calculation with the benefit of reduced memory requirements. Depending on the application we can select the most beneficial method for storing the coefficient factors needed for the FFT computation.

Radix-4 Decimation-in-Time Decomposition

In cases that N is an integer power of 4 ($N = 4^v$) we can further reduce the computational cost of the DFT, by using the *Radix-4* decomposition algorithm. We can rewrite the (3.1) in

terms of four partial sums:

$$\begin{aligned} X[k] &= \sum_{r=0}^{(N/4)-1} x[4r]\cdot W_N^{4rk} + \sum_{r=0}^{(N/4)-1} x[4r+1]\cdot W_N^{(4r+1)k} \\ &\quad + \sum_{r=0}^{(N/4)-1} x[4r+2]\cdot W_N^{(4r+2)k} + \sum_{r=0}^{(N/4)-1} x[4r+3]\cdot W_N^{(4r+3)k} \\ &= \left(\sum_{r=0}^{(N/4)-1} x[4r]\cdot\left(W_N^4\right)^{rk}\right) + W_N^k\cdot\left(\sum_{r=0}^{(N/4)-1} x[4r+1]\cdot\left(W_N^4\right)^{rk}\right) \\ &\quad + W_N^{2k}\cdot\left(\sum_{r=0}^{(N/4)-1} x[4r+2]\cdot\left(W_N^4\right)^{rk}\right) + W_N^{3k}\cdot\left(\sum_{r=0}^{(N/4)-1} x[4r+3]\cdot\left(W_N^4\right)^{rk}\right) \end{aligned} \tag{3.14}$$

and since $W_N^4 = W_{N/4}$ we have:

$$\begin{aligned} X[k] &= \left(\sum_{r=0}^{(N/4)-1} x[4r]\cdot W_{N/4}^{rk}\right) + W_N^k\cdot\left(\sum_{r=0}^{(N/4)-1} x[4r+1]\cdot W_{N/4}^{rk}\right) \\ &\quad + W_N^{2k}\cdot\left(\sum_{r=0}^{(N/4)-1} x[4r+2]\cdot W_{N/4}^{rk}\right) + W_N^{3k}\cdot\left(\sum_{r=0}^{(N/4)-1} x[4r+3]\cdot W_{N/4}^{rk}\right) \\ &= G[k] + W_N^k H[k] + W_N^{2k} Z[k] + W_N^{3k} Y[k] \end{aligned} \tag{3.15}$$

where $G[k]$, $H[k]$, $Z[k]$ and $Y[k]$ are $N/4$-point DFTs. Therefore they are periodic with period $N/4$, and based on the identities $W_N^{N/4} = -j$, $W_N^{N/2} = -1$ and $W_N^{3N/4} = j$ we have:

$$\begin{aligned} X\left[k+\tfrac{N}{4}\right] &= G[k] + W_N^{k+N/4} H[k] + W_N^{2(k+N/4)} Z[k] + W_N^{3(k+N/4)} Y[k] \\ &= G[k] - jW_N^k H[k] - W_N^{2k} Z[k] + jW_N^{3k} Y[k] \end{aligned} \tag{3.16}$$

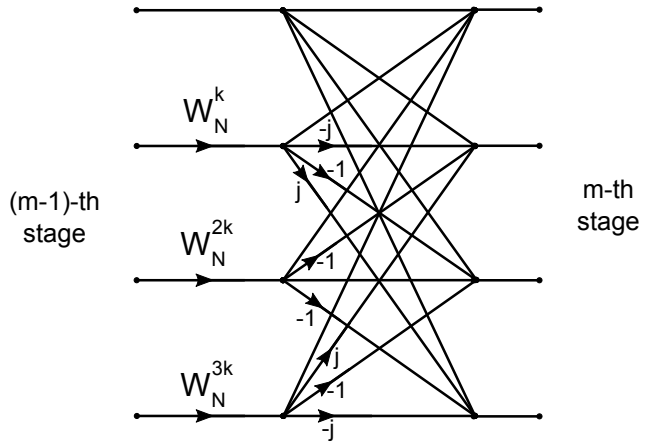


Figure 3.7 Flow graph of the decimation-in-time radix-4 butterfly.

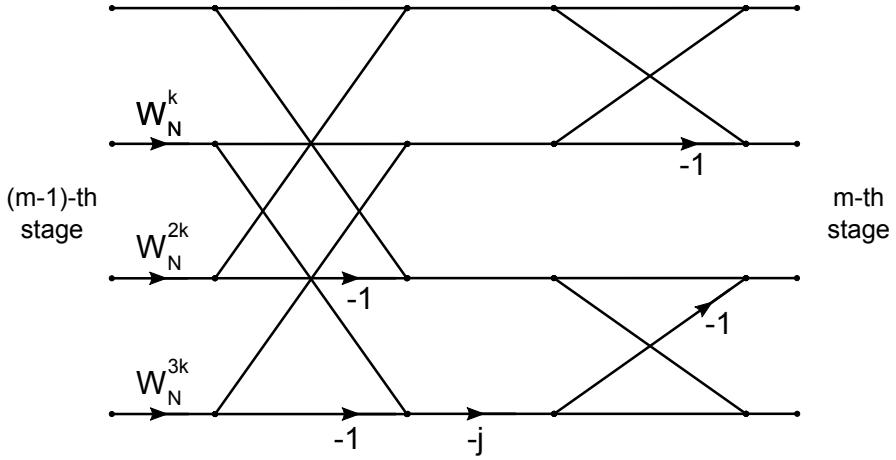


Figure 3.8 Flow graph of the simplified decimation-in-time radix-4 butterfly by using radix-2 butterflies (Radix-2^2 butterfly).

where $k = 0, 1, \ldots, N/4-1$.

Similarly:

$$X\left[k+\tfrac{N}{2}\right] = G[k] - W_N^k H[k] + W_N^{2k} Z[k] - W_N^{3k} Y[k], \quad k = 0, 1, \ldots, N/4-1 \tag{3.17}$$

$$X\left[k+\tfrac{3N}{4}\right] = G[k] + jW_N^k H[k] - W_N^{2k} Z[k] - jW_N^{3k} Y[k], \quad k = 0, 1, \ldots, N/4-1 \tag{3.18}$$

Figure 3.7 shows the flow graph of the Radix-4 Decimation-in-Time butterfly computation. From the flow graph we can see that the Radix-4 DIT butterfly needs three complex

multiplications, twelve complex additions and eight trivial multiplication with specific constant values. We can use Radix-2 butterflies to calculate the Radix-4 computation. Figure 3.8 shows the simplified Radix-4 Decimation-in-Time butterfly, which uses four Radix-2 butterflies to reduce the number of complex additions. This Radix-2^2 butterfly computation needs three complex multiplications, eight complex additions and five trivial multiplications with constant values.

Radix-2^s Decimation-in-Time Decomposition

We can generalized the decomposition technique from the above subsections to construct the Radix-2^s Decimation-in-Time decomposition. Setting $q = 2^s$, a radix-q DIT FFT algorithm may be developed from decomposing (3.1) into q partial sums:

$$\begin{aligned}
X[k] &= \sum_{n=0}^{N-1} x[n] \cdot W_N^{nk} \\
&= \sum_{u=0}^{q-1} \sum_{r=0}^{N/q-1} x[qr+u] \cdot W_N^{k(qr+u)} \\
&= \sum_{u=0}^{q-1} W_N^{uk} \sum_{r=0}^{N/q-1} x[qr+u] \cdot W_N^{k(qr)} \\
&= \sum_{u=0}^{q-1} W_N^{uk} \sum_{r=0}^{N/q-1} x[qr+u] \cdot \left(W_N^q\right)^{rk} \\
&= \sum_{u=0}^{q-1} W_N^{uk} \left(\sum_{r=0}^{N/q-1} x[qr+u] \cdot W_{N/q}^{rk} \right)
\end{aligned} \tag{3.19}$$

Note that (3.5) and (3.15) are special cases of the equation (3.19) when $q = 2$ and $q = 4$. According to (3.19), the time series can be decimated into $q = 2^s$ sets so that each of the q partial sums, can be recursively computed independent of each other. Each of the partial sums represents the DFT of a subproblem of size N/q.

3.2 FFT Architectures

Several FFT architectures for hardware implementation have been proposed in the literature. Each architecture differentiated in terms of the used radix butterfly processing elements and the data management scheme used [246]. The majority of the hardware FFT architectures can be categorized, based mainly on the data management scheme, in the following three groups:

- Fully parallel FFT architectures
- Pipeline/Cascaded FFT architectures
- Memory-Based/Column FFT architectures

A *Fully parallel* or *Array* FFT architecture is a direct mapping of the FFT signal flow graph to hardware. For a N-point FFT fully parallel architecture, $(N/2) \cdot log_2 N$ radix-2 butterfly units are required ($log_2 N$ stages with $N/2$ radix-2 butterfly units each). The data from one stage to the next should be permuted based on the FFT algorithm. This architecture can be pipelined to take N inputs and produce N outputs at each clock cycle [43]. We can implement fully parallel FFT architectures with higher radixes or split/mixed radixes. In such cases the permutation network from one stage to another become extremely complex. As N grows the required processing elements are increased linearly and the routing complexity of the architecture becomes significantly, even in the simple case of radix-2 butterfly units.

The fully parallel architecture of the FFT has increased throughput and low latency compare to other FFT architectures but the significantly area and power overhead can be prohibitive for implementation of FFT processors supporting large numbers of N. Furthermore, these FFT architectures can be used only in applications that all the inputs of the FFT are available at every clock cycle, to maximize the throughput and with only small number of FFT points, to minimize the area and power overhead of the FFT processor.

Another category of FFT hardware architectures is that of the *Pipeline* or *Cascaded* FFT architectures. For the majority of the pipeline architectures there is only one butterfly processing unit for each of the FFT stages. In case of a N-point FFT with radix-r butterfly units there are $log_r N$ stages and processing units. In contrast with the fully parallel architecture, in the case of cascaded architectures there is no need for all the input data of the FFT, to be

available at each clock cycle, to maximize the throughput. For the case of a radix-r pipeline FFT processor, only r input data are needed at every clock-cycle.

At each FFT stage, the butterfly calculations performed in data, with different "distances" between them (e.g. for the case of a radix-2 DIT butterfly unit at the first stage of the FFT calculation, the data entering the butterfly should have "distance" of $N/2$ between them, while for the butterfly unit at the second stage of the FFT, the input data should have "distance" of $N/4$). In a pipeline FFT architecture the correct "distances" between the data elements entering the butterfly unit, at each FFT stage, is guaranteed with the use of memory elements. A cascaded FFT processor can start the calculation of the next FFT frame while processing the previous one (e.g the butterfly unit of the first stage, after processing the last input of the previous frame it can store the first input of the next FFT frame, for the "distance" correction process). This feature makes this architecture favorable for real-time, streaming and high-throughput applications. The pipeline FFT processors can be further sub-categorized, based on the radix and memory architecture used. The various pipeline FFT architectures are presented in the following section.

The *Memory-based* or *Column* FFT architecture utilizes one or more butterfly units for all the FFT stages, while storing the input data and the intermediate results to memories. In the case of a N-point FFT with one radix-r butterfly unit, the memory based FFT processor requires, at least, N/r clock cycles to perform the calculations for one FFT stage and store the intermediate results to memory, assuming that it can read and write r data in parallel from the memory (memory can be splitted in r different banks for parallel access). A total number of $(N/r) \cdot log_r N$ clock cycles and memory requirements of at least N data required, for a memory-based FFT architecture, to perform one FFT calculation. A disadvantage of the memory-based FFT architecture is that it can not process the next FFT frame before the end of the FFT calculations on the current frame. A memory-based FFT processor minimizes the hardware resources needed and can be used in low area/power applications. There are several memory-based FFT architectures that can be used in real-time and streaming applications (Continues-Flow architectures), by using high-radix butterflies, memory buffers and higher clock frequencies (than that of the input data sampling frequency).

A straight forward implementation of a memory-based FFT processor, will include two memories of N data each, one for the inputs of the butterfly unit (outputs of the previous

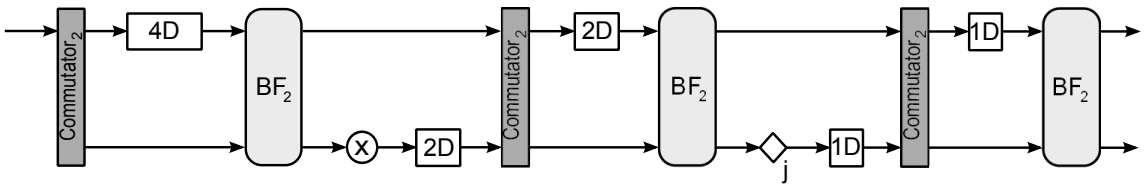


Figure 3.9 Radix-2 DIF MDC Architecture for an 8-point FFT

stage) and one for the butterfly unit outputs (input of the next stage). A more efficient implementation should have only one memory to store N data, and the butterfly unit should write the output data in the same memory positions with the input data (which reads). This requires an addressing scheme based on specific permutations which ensure that the butterfly unit can read and write at the same memory positions, for each of the FFT stages, the correct data for the FFT calculations. These FFT architectures is a sub-category of the memory-based FFT processors and they are known as *In-Place* FFT architectures. The sub-categories of the memory-based FFT architectures are presented in Section 3.2.2.

3.2.1 Pipeline FFT Architectures

Pipelined FFT processors characterized by the continues processing of the input data, which arrive in word sequential format. As mentioned before, the cascaded FFT architecture, utilizes one butterfly unit for each of the FFT stages. The FFT calculation requires that the input data, for each of the butterfly processing elements, should have the correct "distance" between them, based on the FFT length and the FFT stage. Different approaches in "distance" correction mechanism and butterfly radixes used, result in various memory and butterfly utilization factors and different pipeline architecture.

Multi-path Delay Commutator Architectures

The *Multi-path Delay Commutator (MDC)* or *Feed-Forward* architecture [240], utilizes delay elements (memory buffers) and switching circuits to ensure the correct "distance" between the input data of each butterfly processing unit. The input samples are splitted into parallel streams, based on the used radix, and float forward to the butterfly units. Shuffling circuits (commutators) and delay buffers are responsible to "align" the output data from one butterfly (FFT stage) to the next, based on the FFT data flow graph.

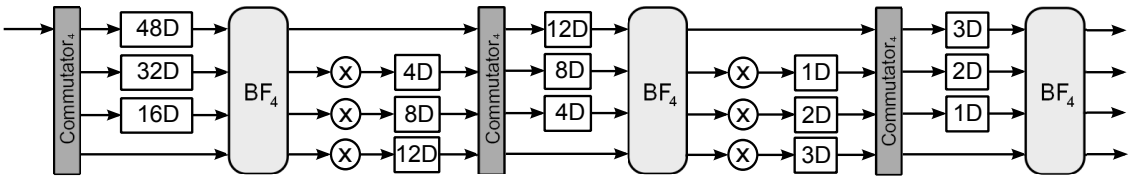


Figure 3.10 Radix-4 DIF MDC Architecture for an 64-point FFT

A radix-2 *Decimation-in-Frequency (DIF)* MDC architecture, for an 8-point FFT, is depicted in Fig. 3.9. The first commutator circuit splits the single input stream to two parallel data streams. The upper data stream is delayed for $N/2$ cycles with the use of a delay buffer, before entering the radix-2 butterfly unit. The second output of the butterfly is multiplied with the appropriate coefficient, before the second delay buffer (delay of $N/4$ cycles). The second commutator rearrange the data streams and the first data stream is delayed for $N/4$ cycles before the second butterfly unit. Again the second output of the butterfly unit is multiplied with the appropriate coefficient, and a delay buffer is used before the last commutator circuit. The last butterfly unit forwards the FFT output in two parallel streams.

The radix-2 DIF MDC architecture utilizes log_2N butterfly elements, log_2N-2 complex multiplier modules and $3N/2-2$ registers for the delay buffers. It has a 50% utilization on both the butterfly elements and the complex multiplier modules [95], [128]. The low utilization factor of the hardware resources and the large number of delay elements are the main disadvantages of the MDC architecture. On the other hand this architecture has a very simple control and can process input samples in parallel streams.

Figure 3.10 shows an 64-point radix-4 DIF MDC architecture. The first commutator splits the input samples into four data streams and delay buffers align the streams before the first radix-4 butterfly unit. The three outputs of the butterfly are multiplied with the appropriate twiddle factors and with the use of delay buffers are forwarded to the next commutator circuit. A similar approach is used for the next two stages and the last radix-4 butterfly element forwards the FFT output data in four parallel streams.

The architecture of Fig. 3.10 utilizes log_4N radix-4 butterfly units, $3log_4N$ complex multiplier modules and $5N/2-4$ registers for the delay buffers. It has a very low utilization factor of 25% for the hardware resources and a very simple control logic. We can increase the utilization factor by processing multiple FFT frames in parallel [95], [128], [40], [352].

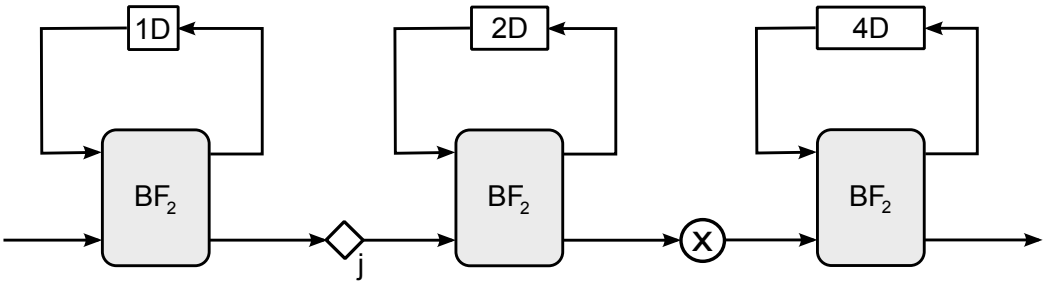


Figure 3.11 Radix-2 DIT SDF Architecture for an 8-point FFT

Furthermore, we can increase the throughput of the MDC-FFT processor by using high-radix butterflies [251], [130], [4], [78] or mixed-radix architectures [359], [152].

Single-path Delay Feedback Architecture

The *Single-path Delay Feedback* architecture [86], [334], has a more efficient way of using the registers, by storing some of the butterfly outputs in feedback shift registers. Only a single data stream goes through the multiplier at every stage, regardless of the selected radix implementation. It has the same number of butterfly units as in the MDC approach, but with reduced memory requirements.

A radix-2 DIT SDF architecture for an 8-point FFT, is depicted in Fig. 3.11, assuming that the input samples are in bit-reversed order. The upper output of the butterfly at the first stage, is stored in a feedback shift register of size "1" and it can be used as input in the same butterfly with a 3rd input sample (delay of 1 clock cycle). The other output of the first butterfly is forwarded to the multiplier of the next stage of the FFT calculation. The second stage of the SDF FFT architecture has a similar butterfly but the feedback shift register has the double size of that of the first stage. Each of the FFT stages, of the radix-2 DIT SDF architecture, has a feedback shift register of double size than that of the previous stage.

The radix-2 DIT SDF architecture utilizes $log_2 N$ butterfly elements, $log 2N-2$ complex multipliers and only $N-1$ registers for the feedback buffers, compared to $3N/2-2$ registers of the MDC architecture. The memory requirements for the SDF architecture is minimal [96]. The utilization of the butterfly and the complex multiplier module is the same as in the MDC architecture (50%) while the control is more complex.

Figure 3.12 shows a 64-point radix-4 DIT SDF architecture. At the first stage, three out of four butterfly outputs are stored in three feedback shift registers of size "1" each. Only

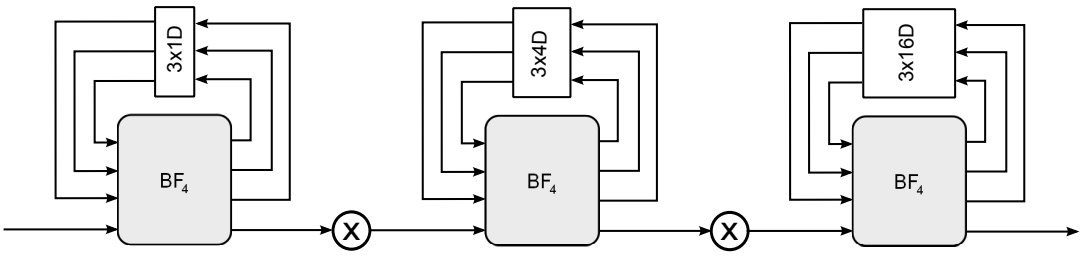


Figure 3.12 Radix-4 DIT SDF Architecture for an 64-point FFT

one of the butterfly outputs is forwarded to the next stage. Each of the FFT stages has four times more registers, for the feedback shift register units, than that of the previous stage, for each of the three butterfly outputs.

The radix-4 SDF FFT architecture utilizes $log4N$ radix-4 butterfly processors but only $log4N-1$ complex multiplier modules, compared to the $3log4N$ complex multipliers of the MDC architecture. The utilization of the complex multiplier modules is increased to 75% compared to 25% of the radix-4 MDC architecture, but the utilization of the more complex butterfly unit is only 25% [97]. Furthermore, the memory requirements of the radix-4 SDF architecture are the same as in the radix-2 SDF architecture ($N-1$ registers), while the radix-4 MDC FFT architecture requires $5N/2-4$ registers. Finally, the control of the radix-4 SDF architecture is considerably more complex than that of the radix-4 MDC architecture [21]. We can increase the hardware utilization of the SDF FFT architectures by using high-radix [72], [21], [95] [96], [131], [97] split-radix [356], [153] or mixed-radix implementations [348], [334], [45] or by using parallel memory structures [54].

Multi-path Delay Feedback Architectures

The *Multi-path Delay Feedback (MDF)* or *Parallel Feedback* FFT architectures, extends the SDF FFT architecture for high-throughput applications, by using parallel paths (MDC architecture) and feedback loops (SDF architecture) [316], [273], [151], [158]. In contrast to SDF architecture, parallel input streams can be handled by multiple butterfly modules. Each of the butterfly units, in the same FFT stage, has a separate delay shift register structure and complex multiplier module. Additional delay registers and shuffling circuits are needed (Data Commutators) between the FFT stages [41], [51], [294]. In general, the MDF FFT architecture contains multiple interconnected SDF paths, and each path is responsible for

managing one of the parallel input streams of data.

This architecture has an efficient use of memory resources, but suffers of small utilization for the arithmetic units, which is a common problem of feedback FFT architectures. We can decrease the number of complex multiplication modules needed, by using radix-2^i butterfly structures [41], [175], [151], [158], [51], [273] or mixed-radix architectures [314], [49], [172], [161]. By reordering the input data samples and rescheduling the timing of the complex multiplications we can increase the utilization of the hardware resources [174]. A mixed DIF/DIT flavor of the MDF architecture is proposed in [316], where higher utilization of the arithmetic modules is achieved with an increase of the number of complex multipliers and adders. A substitution of the complex multipliers with shift-add multipliers in [176], has as result a higher utilization of the multiplication units.

The flexible radix configuration MDF architecture in [293] supports variable-length FFTs and multiple input streams (MIMO), with improved hardware efficiency and power consumption. A reduction on constant twiddle factor multiplication modules, by relocating and sharing the units on different processing paths, is proposed in [353]. In a single-path pipeline FFT architecture (such as SDF architectures) the process of bit-reversing either the input data samples (DIT), or the output FFT data (DIF) is performed by a reordering circuit with low complexity. For architectures with multiple parallel processing paths (such as MDC and MDF), the task of bit-reversal is more complex. In [359] the reordering circuit is part of the input/output commutator, while in [352] a FIFO-like memory structure is performed the reordering of the FTT output data. Finally, a generic parallel bit-reversing scheme, for MDC and MDF FFT architectures is proposed in [46], based on single-port memories and area efficient architecture.

Single-path Delay Commutator Architectures

The *Single-path Delay Commutator (SDC)* architecture is an Single-path pipeline architecture, based on the SDF, which uses data commutators to minimize the number of arithmetic units, while increasing their utilization [321]. It uses delay elements to reorder the data, in each stage, to achieve better utilization with reduced hardware resources. The number of needed complex multipliers are the same as in the SDF architecture, while it needs less complex adders and more memory elements [40], [179]. The control logic of the SDC

architecture is more complex, compared to that of the SDF or MDC architecture.

By changing the reordering on the last data commutators of an SDC architecture, we can reduce the complexity of the bit-reversal circuit at the output of the FFT calculation [40], [179]. Furthermore, we can combine the SDC and SDF architectures, in a hybrid pipeline architecture, with reduced arithmetic units, increased hardware utilization and reduced memory requirements for the bit-reversal circuit [321], [317]. This hybrid pipeline architecture has optimized usage for the delay elements and reduced power consumption, without any cost on the FFT throughput [317].

The SDC pipeline architecture can be used with high-radix butterflies to further reduce the number of arithmetic units with more complex butterfly structures [22], [317]. A matrix decomposition of the FFT can result in a generic matricial expression for efficient radix-r^k SDF or SDC pipeline architectures [61]. This method can be used for efficient pipeline architectures, for large FFT calculations with high radix butterflies.

3.2.2 Memory-based FFT Architectures

All the pipeline FFT architectures use the memory elements only for the re-arrange of input data and intermediate results, to correct the "distance" of butterfly inputs at every FFT stage. Each of the FFT stages has each own processing unit and the results from one stage "flows" to the next, without any need for storage. This architecture maximize the FFT throughput with an impact on the hardware resources requirements. For large FFT lengths the pipeline architecture may not be suitable, due to high hardware resources demands. The memory-based FFT architectures can balance the trade-off between hardware resources and throughput performance, where one processing element is usually used to compute all the butterfly operations of the FFT calculation, while the intermediate results are stored to memory elements, from one stage to the next. Therefore, these architectures are more suitable for low-area and low-power applications, than the pipeline FFT architectures [119], [90].

A memory-based FFT architecture needs at least N/r clock-cycles to perform all the calculations for one FFT stage, assuming a single radix-r butterfly processor and r memory banks, for parallel memory accesses. A total processing latency of at least $(N/r) \cdot log_r N$ clock-cycles is needed for a radix-r memory-based FFT architecture to perform one FFT calculation. We can reduce the total processing time by using multiple butterfly processors.

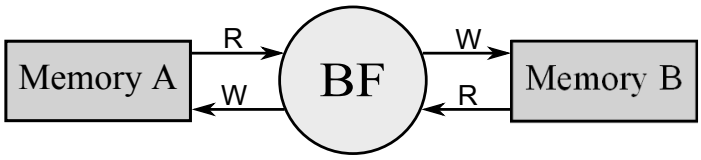
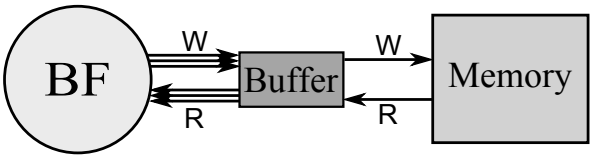
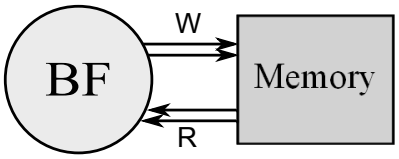


Figure 3.13 Dual Memory FFT Architecture



(a) Buffer/Cached-based In-place (single memory) FFT architectures



(b) Conflict-free In-place (single memory) FFT architectures

Figure 3.14 Single Memory (In-place) FFT architectures

An architecture with k radix-r butterfly units, has a total processing latency of at least $(\frac{N}{rk}) \cdot log_r N$ clock-cycles, but it needs rk memory banks for parallel accesses. For streaming applications the memory-based FFT architectures, need additional memories to store the input data of the next FFT frame, while processing the current FFT frame, and to forward the previous FFT frame. There are several techniques to reduce the storage requirements of these *Continuous-flow* memory-based FFT architectures [11], [241].

The memory-based FFT architectures can be categorized into three groups, based mainly on the memory hierarchy used. The *Dual-Memory* FFT architectures use two memories, of size N, for the intermediate results of the FFT calculation, as shown in the Fig 3.13. For the first FFT stage the butterfly unit reads data from *Memory A* and stores the outputs to *Memory B*. At the second FFT stage the butterfly processor reads data from *Memory B* (outputs of first FFT stage) and writes the results to *Memory A*. At each of the FFT stages the read and write memories are swap with simple muxing logic. This architecture has a simple control logic and it can be implemented with single-port memories. The memory addressing scheme has reduced complexity but the specific architecture has increased data storage requirements and a low memory utilization factor [69], [121].

The *Single-Memory* or *In-Place* FFT architectures have low storage requirements by using an "in-place" addressing scheme, where output data are written to the same memory locations from where input data are read, for each of the FFT stages (butterfly calculations),

as shown in Fig. 3.14b. These architectures have a complex control and memory addressing logic. The majority of these architectures need dual-port memories for parallel read/write operations (low-latency architectures) with an increased memory utilization factor, compared to Dual-Memory architectures [59], [181], [278]. Finally, there are the *Buffer-Memory* or *Cache-Memory* FFT architectures, which use intermediate buffers or cache memory to store butterfly results, as shown in Fig. 3.14a. This cache-based operation can be used in In-Place architectures, either to eliminate the memory address conflict problems or to reduce the power consumption by using a combination of single-port memories with dual-port cache/buffer memories [36], [143], [10], [142], [173]. Another solution for the memory address conflicts is the use of conflict-free addressing schemes for in-place architectures, which increase the circuit complexity [189], [127], [290], [228].

Buffer/Cache-based In-Place FFT Architectures

The main concept of the cached FFT architectures, is to reduce the number of main memory accesses, by using a small cache memory closer to the processing datapath. For every FFT stage all N data should be read from memory, processed by the butterfly unit and written back to the memory. At the next FFT stage the same data (results from previous stage) should be read, processed and written back to the memory. A cached FFT architecture uses an additional smaller storage unit, to perform more than one stage calculations on a small portion of the data [10], [142]. This technique results to reduced main memory accesses with the overhead of a small additional storage element. The power consumption of the memories on a memory-based FFT architecture is a large portion of the power dissipation of the FFT processor [163]. By using a small additional memory, cached FFT architecture reduces the accesses on the large main memories resulting in lower power consumption for the FFT processor.

The first cache-based FFT algorithms, consider the computation of a large FFT using multi-processor systems with distributed memories, when the total main memory was smaller than the FFT dataset. These algorithms split the input data (stored on external storage) in smaller portions and perform several small FFT calculations with intermediate transposition steps on the data. The key idea was to minimize the use of the slow external storage unit for intermediate FFT results. The “Four-step” FFT algorithm is quite effective

in reducing the number of passes through the dataset by using buffers smaller than N, while increasing the arithmetic computations [3], [288], [80]. These algorithms are natural order FFT calculations and there is no need for any bit-reversing of the input or output data. The "Six-step" and "Two-pass" algorithms reduce further the number of passes through the dataset to two, while using larger buffers, more arithmetic operations and increased memory accesses [13], [289]. These algorithms are used mainly on software implementations, for the FFT computation on multi-processor systems with shared/distributed memory, but they can used also on hardware implementations, when the total main memory of the FFT processor is limited and smaller than N.

As mentioned above, the cached-based FFT architectures are mainly used to reduce the power consumption by minimizing the memory accesses, on the FFT processor. A *ping-pong* cache scheme of two 32 words buffers [9] is used on the memory-based FFT architecture of [10], to reduce the main memory traffic by a factor of 5, on a 1024-point FFT processor. A variable length pipeline of 9 or 10 stages on the butterfly unit is used to solve the memory access conflict problem and handle the read-after-write data hazard on the cache buffers. A similar *ping-pong* cache scheme, with two 64 words caches, is used on the dual-stream (MIMO 2x2), continuous-flow FFT architecture of [48]. A mixed-radix butterfly, supporting radix-$2^3/2^2/2$ configurations is used for variable-length FFT ($128 \sim 2048$ points) support, while a simple control circuits for the cache buffers saves the half of the main memory accesses. A variable-length FFT processor based on radix-2 butterfly is presented in [142] and [143]. By merging three radix-2 stages in one *super stage* and using a small cache memory, the main memory accesses are reduced. Additional permutations on the output data of each super stage are required for the FFT calculation. Finally, a three level cache scheme is used in [173], to minimize the power consumption of an 8192-point FFT processor. The main memory banks and the second level, prefetch buffer cache, can be implemented with low power single-port memories, while a third level buffer cache is used with a three-step radix-8 butterfly to reduce the processing latency of the FFT processor.

Conflict-Free In-Place FFT Architectures

As mentioned above the *Single-Memory* FFT architectures with conflict-free addressing schemes reduce the storage requirements of the FFT processor, to a minimum of N words,

for the FFT calculation. These architectures [181],[308] store the butterfly results in the same memory locations as the butterfly inputs without the need of an intermediate buffer (*Cache-Memory* architectures). A radix-r butterfly unit has r input and r output data ports. All the input data should be read simultaneously from the memory for increased throughput. The parallel memory access on single-memory architectures is performed by implementing the storage unit with several memory banks. For r parallel accesses we should have at least r memory banks, with N/r words capacity each [278], [127], [290]. For each butterfly operation we should read r data from r memory banks and write r data back to the storage unit. A conflict-free addressing scheme should ensure that all r input data of the butterfly can be read simultaneously from the storage unit and all r output data can be stored back to the memory. This property should be hold for all butterfly operations at each of the FFT calculation stages.

Pease in [230] observed that the address parities of butterfly inputs are different, and he proposed a scheme to segment the memory into several banks using this property to meet the memory access requirement. Cohen in [59] proposed a simple addressing scheme for a radix-2 FFT processor with two memory banks. This addressing scheme is based on the Pease's observation, and parity calculations are needed for the selection of the memory bank. Furthermore, there is the need for interchanging circuits for data addresses and butterfly input and output data. Johnson in [127] proposed a generic radix-r conflict-free addressing scheme, with r memory banks. The address generation circuit includes a tree of modulo-r adders and the bank selection performed with barrel-shifters. The high complexity of the addressing scheme grows logarithmically in the transform size N.

Ma in [189] proposed a more simple addressing scheme, for the case of radix-2 FFT processor, which includes barrel-shifters for the address generation and swapping circuits for the addresses and data ports of the memory banks. Furthermore, four additional registers are needed, for the memory address and data ports, to avoid conflicts on the write operations. In [190] Ma and Wanhammar proposed a similar addressing scheme with four memory banks for radix-2 FFT processors. The new addressing scheme does not require the additional registers and swapping circuit for the memory address port and the additional registers for the memory data port. A reordering of the sequence of the butterfly calculations and a memory-cache architecture for the twiddle factors result in reduced RAM and ROM

accesses and power consumption, for the FFT processor. Parallel memory addressing in FFT processors with multiple butterfly units is proposed in [100]. The access conflicts are avoided by reordering the operands with the use of an interconnection network. The reordering is performed in time and space, which requires additional operand registers. Takala in [290] proposed a generic radix-r addressing scheme for parallel memory access on a multi-butterfly FFT processor. In comparison with Johnson's addressing scheme [127] the new address generation requires only rotation and XOR calculations on a subset of the address bits, while supporting multiple butterfly units.

Xiao et al. [345], propose a heuristic approach for the radix-2 FFT algorithm. A modified butterfly unit, with exchange circuits for both input and output ports, is used for reordering the FFT processing. The conflict-free in-place address generation does not require any addition steps and is based on counters, shifters and delay elements. Compared to Ma's scheme [189] this scheme has reduced complexity and delay, but can be used only for radix-2 FFT architectures. A radix-4 conflict-free scheme with four memory banks is presented by Xiao et al., in [346]. Similar to [345], this heuristic approach does not require addition steps for the address generation and has barrel-shifters for the exchange operations on the butterfly input and output ports. Furthermore, this scheme requires four registers for each of the four input and output ports of the radix-4 butterfly (32 registers in total), and additional barrel-shifters for these data-path registers, to avoid memory conflicts. The address generation circuit is based on counters and barrel-shifters. A total number of $2r^2$ data-path registers, $4r$ multiplexers ($4r$ to 1) and the address generation circuit are require for a radix-r implementation of the FFT processor.

Authors in [247] present a mathematical proof that there exists at least one set of permutations that can be used to resolve the memory conflicts. Furthermore, they prove that these permutations result in a correct FFT algorithm, for arbitrary radix decomposition, and can be used for parallel memory access in both pipeline and in-place FFT architectures. The presented technique permutes the addresses of the elements prior to memory write operations of each stage and uses the inverse permutation for reading from the memory in the following stage. The advantages of this technique, compared to Ma's scheme [189], are the efficient extension to arbitrary radix decompositions of the FFT and the elimination of the requirement for any kind of auxiliary queue registers for the results of the butterfly

processor.

In [209], authors propose a parallel accessing scheme for radix-2 FFT processors, with two memory banks, based on permutations. A mathematical proof is presented that the specific permutation provides the correct FFT calculations and that it resolves the memory conflicts. The proposed technique applies a specific permutation to the output data of each butterfly, so that the data of any transformation pair at the following stage will be located in different memory banks. The delay for the address generation circuit is that of a counter and a single XOR gate compared to that of a counter and an adder in [127], a counter, a shifter and three muxes in [345] and to the LUTs in [247]. Furthermore, the presented technique in [209], eliminates the butterfly output queues [189] and it also results in a regular addressing during all the FFT stages compared to [345]. A radix-2 FFT processor with parallel data access and a single memory bank implementation is proposed in [236]. Authors, present a specific permutation, which allows the FFT data to be stored as pairs in a single memory with $N/2$ locations. The butterfly implementation includes 4 input and 4 output data registers and multiplexers to perform the permutations on data ports. The address generation circuit includes a simple circular counter and a specific permutation must be applied only for the first stage of the FFT calculations.

As mentioned above the in-place FFT architectures use one memory buffer of size N, for the butterfly intermediate results. In a streaming application the data for the next FFT frame should be stored in a separate memory buffer, while the FFT processor computes the current FFT frame. Another memory buffer is needed to output the previous FFT frame. Conclusively, a straight-forward implementation of an in-place continuous-flow FFT processor, requires 3 memory buffers of size N [308]. In [241], authors propose a radix-2 in-place FFT architecture, in which Decimation In Time (DIT) and Decimation in Frequency (DIF) decompositions are used for concurrent FFT frames. This technique reduces the memory requirements of the continuous-flow FFT processor, to two main memories of size N, one for the FFT calculations (computation buffer) and one for the I/O operations (I/O buffer). A new FFT frame can be written to the I/O buffer while the previous FFT frame is computed (using the computation buffer), and the twice previous frame is read out from I/O buffer. The key idea is that for the two FFT frames that must be written to and read out from the I/O buffer a different decomposition is used, so the same memory addresses can be used

for write and read operations: while the symbol k is read out from I/O buffer in bit-reversed order (DIF decomposition) the symbol $k+2$ can be written in bit-reversed order, to the same memory locations, and it should be processed with the DIT decomposition.

Authors in [11] present a mixed radix-4/2 in-place FFT architecture, based on the conflict-free addressing scheme of [127]. Exchange circuits on the input and output ports of the butterfly are used, on specific FFT stages, to avoid memory conflicts and to ensure continuous flow operation with only two memory modules. Furthermore, a more complex addressing scheme is adopted, in comparison with that of [127], to support the mixed radix-4/2 architecture. A similar continuous-flow approach is presented in [125] with conflict-free addressing scheme based on tree of modulo-4 adders. Moreover, this approach is used in [112] for a variable-size, mixed radix-2/4 FFT processor with continuous-flow operation. A different approach based on radix-2^q MDC units in presented in [301] and [344]. Authors propose a generic conflict-free addressing scheme for FFT architectures with multiple radix-2^q MDC butterfly units. This approach supports variable-size FFT calculations with mixed-radix implementations and continuous-flow operation with $2N$ words storage requirements.

Chapter 4

FFT Architectures for MIMO Systems

4.1 Introduction

The performance of the Fast Fourier Transform (FFT) and the Inverse FFT (IFFT) algorithms plays a crucial role in emerging wireless technology standards that are based on Orthogonal Frequency Division Multiplexing (OFDM). There are several standards that use OFDM as a core function in their baseband processing including the IEEE 802.11 (WiFi), IEEE 802.16 (WiMAX), and the 3G Long Term Evolution (LTE) standard which has emerged as a comprehensive evolution of the Universal Mobile Telecommunications System (UMTS). These standards support high-data rates that require executing FFT/IFFT at relatively high speeds, while cost constraints imply the use of minimal resources. The problem of achieving high throughput rates leads to a pipelined execution by using a processor for each FFT stage. Single datapath Delay Feedback (SDF) architectures [298] and variations proposed for OFDM systems [282], [97], [96], [95] have been considered as the most appropriate solution because, apart their pipelined structure, they need minimal memory volume for data storing.

The majority of the new standards for wireless communications, supports Multiple-Input Multiple-Output (MIMO) for higher data rates, better quality of service and increased network capacity and spectral efficiency. The core idea behind MIMO is that signals sampled in the spatial domain at both ends are combined is such a way that they either create effective multiple parallel spatial data pipes (therefore increasing the data rate), and/or add diversity

to improve the quality (bit-error rate) of the communication [302], [274], [141]. However, the MIMO-OFDM systems has increased computational complexity, for the baseband processing, compared to the Single-Input Single-Output (SISO) OFDM systems, due to the parallel data streams. Furthermore, several bandwidths should be supported for each protocol resulting in increased computational complexity and control. A baseline MIMO-OFDM system has multiple transmitter/receiver baseband chains, one for each of the MIMO data streams. This results in demanding resource requirements mainly because same blocks, with high computational complexity are used in the multiple baseband processing paths [233], [284]. In such systems adopting the SDF solution, for the FFT processor architecture, leads to a large number of multiplication processing units, which increase the complexity and the power consumption of the system.

4.2 FFT Schemes for MIMO Systems

The demanding computation of the FFT and IFFT is part of every OFDM system. Multiple bandwidths support in several protocols, results in FFT computations, in the baseband processing, with several FFT lengths. In the concept of MIMO-OFDM systems multiple FFT computations should be performed for the parallel data streams. Several FFT architectures has been proposed in the literature, which focus in the resource sharing between the multiple data streams, in a MIMO-OFDM system, to reduce the hardware requirements, compared to multiple FFT processors. Various FFT architectures also supports variable-length FFT computations for the case of multiple bandwidths.

4.2.1 SDF-based Architectures

The authors of [260] propose an FFT architecture in which several MIMO data streams can be processed by fewer FFT processors which are running at higher clock frequencies, than that of the sampling frequency of the system. A round-robin scheduling with input and output buffers is responsible to handle the FFT computations of multiple data streams, with one or more FFT processors. A single radix-2^4 SDF architecture is proposed for the computation of 2048-point FFTs on a 4x4 MIMO-OFDM system. In [226] and [227] authors propose a similar scheduling scheme with reduced memory requirements for the I/O buffers.

Both scheduling schemes introduce the resource sharing between the MIMO streams, in the module level and have the disadvantage of restricted scalability in terms of MIMO streams and operational frequency of the FFT processor.

Several FFT architectures has been proposed to support multiple data streams. In [361], a mixed-radix SDF-based architecture is proposed with an efficient data-flow scheduling scheme. The proposed processor computes 256-point FFT for two parallel data streams, to support 2x2 MIMO-OFDM systems. The specific architecture is based on radix-2 butterflies and single-port memories, for the delay buffers, operating at double frequency than that of the input sampling. The data-flow scheduling exploits the "idle" time of each butterfly for the calculations of the second data stream. Another SDF architecture for multiple data streams is proposed in [108]. The FFT processor is based on radix-4^2 and radix-$4/2$ butterflies to support variable-length FFT calculations ($128 \sim 2048$ points) for 4x4 MIMO-OFDM systems (e.g. WiMAX, LTE). Authors, use the 75% "idle" time of the radix-4 butterfly for the processing of the multiple data streams, while changes in the data flow and associated memory access improve the storage utilization. This memory re-allocation reduces the feed-back memory requirements and results in an efficient sharing of the multiplier units, between the multiple data sequences.

A generic framework for automatic FFT IP generation for SISO/MIMO OFDM systems is presented in [300]. The FFT architecture for the SISO (Single-Input Single-Output) case is SDF based mixed-radix decomposition with radix-$2/2^2/2^3$ butterfly processors. The selected architecture for the case of multiple data sequences, is a hybrid solution based on SDF and MDC pipeline architectures. Based on the number of MIMO channels the first section of the FFT processor, is based on the specific high radix butterflies to exploit the parallel paths of the MDC architecture, while the last section of the processor in based on parallel small radix SDF sub-FFT processors, to reduce the memory requirements. A heuristic approach is used to determine the data-path bit-width at each stage to increase the fix-point performance of the processor without compromising the complexity of the architecture.

4.2.2 MDC-based Architectures

The multi-path structure of the MDC pipeline FFT architectures is attractive for parallel data stream processing. In [152] a multi-channel mixed-radix MDC architecture is proposed for

FFT processing on MIMO-OFDM systems. The mixed-radix implementation reduces the number of non-trivial multipliers, while extra delay commutator units are introduced for the appropriate alignment of the multiple data sequences. A similar mixed-radix MDC architecture, based on radix-$4/2$ butterflies is proposed in [133] and [76] for 4x4 MIMO-OFDM systems, with variable symbol lengths (64/128). The extra, low-complexity commutator units in [133] result in an efficient sharing of the resources between multiple data streams. In [76] a memory-based reordering unit interleaves the data sequences for an improved sharing of the butterflies and multiplier units.

Authors in [359] propose an area and power efficient FFT processor, for 8x8 MIMO-OFDM systems, based on the mixed-radix MDC architecture. Two radix-8 butterflies are used with several parallel radix-2 processors, to support 128-point FFT computations. The delay commutator units are replaced with I/O buffers and additional address generator circuits, implementing the pre- and post- commutator logic, are used to align the eight data sequences. A similar architecture is proposed in [269] with optimized delay commutator units. A memory array transpose unit is used for the proper alignment of the parallel input sequences and a complex delay commutator is used for the bit-reversing of the output data, instead of the two I/O buffers of [359]. The radix-8 processor of [359] is implemented with radix-2^3 decomposition to reduce the butterfly computational complexity, while the radix-4 booth encoding multiplier is used to improve the circuit complexity of the complex multiplier unit.

A variable-length FFT processor based on mixed-radix (radix-$2/4$) MDC pipeline architecture, for 4x4 MIMO-OFDM systems with variable symbol-length ($64 \sim 2048$), is proposed in [349]. A data mapping module is responsible to align the multiple input data sequences, by using several delay elements and switches, and forward the data to specific butterflies based on the selected FFT length. The proposed mixed-radix decomposition method reduces the number of non-trivial complex multiplication units, compared to radix-2 SDF and MDC architectures. Researchers at [352] propose a mixed-radix MDC pipeline FFT architecture with memory scheduling and input data shuffling to increase the butterflies utilization. The architecture is based on radix-$4/8$ butterfly processors, including a memory-based input data shuffling unit which align the multiple data sequences appropriately before the FFT computations. The four radix-4 stages includes FIFO-based commu-

tators with switching networks (barrel-shifters), while the last module is responsible to unshuffle the output data, with several FIFOs and complex switching networks. A total amount of $(5+\frac{1}{4})N$ memory storage is required for the support of 4x4 MIMO-OFDM systems with variable symbol length ($128 \sim 2048$).

Authors in [118] propose an area-efficient FFT processor for MIMO-OFDM Software Defined Radio (SDR) systems. It support up to 4 MIMO channels with variable length ($64 \sim 2048$ and 1536), for WiFi (IEEE 802.11n), WiMAX (IEEE 802.16e) and 3GPP LTE systems. It is based on a mixed-radix MDC pipeline architecture with radix-3, radix-2 and radix-4 butterfly processors. A FIFO-based data mapping module is responsible for the alignment of the input data sequences, by using commutator logic, while a data re-ordering module re-arrange the output data streams. The proposed architecture has reduced area and memory requirements, compared to radix-$2/3$ SDF and MDC pipeline architectures, while on the other hand, it can not support concurrent multi-protocol operation.

4.2.3 MDF-based Architectures

The parallel data sequences of a MIMO-OFDM system can be handled also by an MDF-based pipeline FFT architecture. An unfolding mixed-radix MDF FFT processor, based on radix-$2/2^3$ butterfly structures is proposed in [172] and [171]. The specific architecture supports up to 4 data streams and variable symbol lengths (64/128) and it is suitable for WiFi (IEEE 802.11n) MIMO-OFDM systems. The multiple input data streams are reordered and grouped, before the first stage of the FFT computations, by a commutator-like module. The first processing unit includes four parallel radix-2 butterfly processors, while the second and third units contains four parallel radix-2^3 processors. All the processing modules include several FIFOs to align the butterfly inputs, while there is an extra module for the re-ordering of the output data. The high radix implementation results in reduction of the complex multipliers, while the data scheduling and delay feedback approach reduces the memory requirements of the FFT processor.

A similar mixed-radix MDF pipeline FFT architecture is presented in [175]. Researchers propose a high-radix decomposition based on radix-2^4 to reduce the number of non-trivial complex multiplication units in the design. The specific FFT processor support up to 4 MIMO channels, with four parallel processing paths, and variable-length symbols (64/128)

to support WiFi (IEEE 802.11n) MIMO-OFDM systems. The high-radix implementation results in more complex control logic for the processor, while a four-parallel Booth multiplier module is introduced to reduce the complexity of non-trivial complex multiplication units. Authors in [162], propose a mixed-radix MDF architecture based on radix-4 and radix-$2/2^2$ structures to support up to 4 parallel data streams and variable FFT lengths (128/256) for WiMAX (IEEE 802.11e) MIMO-ODFM systems. The first pipeline stage in constructed with four parallel radix-$2/2^2$ butterfly units, while the remaining three stages include four parallel radix-4 processors. Several commutators and memory elements are used on each of the stages, to support the four parallel data sequences, while extra modules are responsible to re-order the data streams at the input and the output of the FFT processor.

4.2.4 Memory-based Architectures

Memory-based FFT architectures, while targeting low area and low power applications, they considered not suitable for high throughput implementations, such as MIMO-OFDM systems. Nevertheless, high-radix and/or parallel butterfly implementations, of memory-based FFT processors, have increased throughput, while having reduced hardware complexity and power consumption. Moreover, the majority of the pipeline architectures which supports multiple data streams, have the disadvantage of interleaving the data in a FFT-frame basis and forwarding the MIMO streams sequentially (first stream FFT-frame, second stream FFT-frame, etc). The modules which process the IFFT/FFT processor outputs, are mainly implemented to support parallel data streams as inputs (MIMO support) and not to support multiple MIMO channels sequentially. The memory-based FFT architectures can easily support the parallel data streams output, due to the fact that all the output data are stored in memories. Finally, the specific FFT architecture can easily support variable-length FFT computations for multi-bandwidth and/or multi-protocol SDR applications.

A cached-based FFT architecture, which supports variable-length ($128 \sim 2048$) FFT processing for a 2x2 MIMO system, is presented in [48]. One mixed-radix butterfly processor, capable of processing one radix-2^3, two radix-2^2, or four radix-2 is used. The proposed *ping-pong* cache memory scheme supports two radix-2^3 stages processing with the half accesses, on the main memory, with the use of a 64-words cache storage unit. A continuous-flow architecture, based on [125] is proposed to reduce the memory requirements, while

the two 4K words memories are implemented with eight banks each for parallel access. A block-scaling technique is used to increase the SQNR performance of the FFT processor, with reduced data-path bit-widths. The proposed architecture supports the computation of two 2048-point FFTs within 2052 cycles, supporting 2x2 MIMO WiMAX systems, with operational frequency same as the sampling rate of the system. On the other hand this architecture has limited scalability, to support more than two multiple data streams, while the control of the continuous-flow operations and cache memory handling is complex. Furthermore, the mixed-radix, single cycle butterfly processor has a "long" critical path, resulting in increased power consumption.

A similar FFT architecture is proposed in [109]. The same mixed-radix (radix-$2^3/2^2/2$) butterfly processor and *ping-pong* cache scheme is used to support variable-length ($1024 \sim 8192$) FFT computations for up to 4 MIMO channels (only for the case of 1024 and 2048-point FFTs), with increased operational frequency. The same continuous flow technique [125] is adopted, while two 8K words main memories with eight banks each are used as calculation and I/O buffers. A more efficient block-scaling method is proposed to increase the SQNR performance of the FFT processor, while various FFT length and MIMO channel combinations are supported. The proposed architecture is more flexible and supports both long FFT computations with single data stream or shorter FFT calculations with multiple data streams and increased operational frequency, than the sampling rate of the system. On the other hand, the computational complexity of the continuous flow operations, on the main memories, is increased for multi-channel configurations. In these cases several data, from multiple streams, should be written on the same memory bank at the same clock cycle, for further processing by the FFT processor, while multiple previous results should be read from the same bank to construct the FFT output. Authors, do not consider the computational complexity of a complex scheduling scheme and/or intermediate buffers which should be used to avoid memory bank conflicts, with the specific continuous-flow technique. Alternately, 8 banks per MIMO channel should be used with a reduced complexity addressing scheme for the continuous-flow operation.

The memory-based architectures in [48] and [109] are high-radix/high-throughput FFT processors which are able to process multiple FFT frames in a single FFT frame sampling latency, which can be used to support multiple data streams, in a time-interleaved basis.

This technique can be used with several high-throughput memory-based FFT processors. The radix-2^2 reconfigurable FFT processor in [335] can support variable-length ($16 \sim 128$) computations, with a low-power memory-based implementation. Multiple data streams can be easily supported with proper I/O buffering and increased operational frequency. The mixed-radix (radix-$4/2$), continuous-flow memory-based FFT architecture in [112] can be used for multiple data streams, with a reduced I/O buffering and increased operational frequency. Finally, the optimized radix-16 continuous-flow FFT architecture in [106] can be used for parallel data streams processing, due to the high throughput implementation. The normal-order I/O buffering of the FFT processor results in reduced complexity of the buffering scheme for a multi-channel MIMO support.

High-radix implementations result in high-throughput FFT architectures, with the disadvantage of more complex control and addressing schemes. The increased computational complexity of the high-radix butterflies also affects the critical path of the FFT processor and the needed pipeline stages inside the butterfly module, result in increased latency for each of the FFT stages. The support of multiple data streams further increases, the hardware complexity of the I/O buffering, while the requirement of regular MIMO data-ordering at the input and output of the FFT processor, can result in extreme storage requirements and very complex interconnection networks. The majority of the FFT architectures does not consider the requirement of the regular data-ordering for the case of multiple data streams support. This requirement is essential for an FFT processor which is a part of a pipeline processing path, and without proper data ordering these architectures can only be used with additional memory structures which result in increased hardware complexity and power dissipation for the pipeline processing path.

4.3 Efficient and Scalable In-Place FFT Architecture for SDR/MIMO Systems

This section presents a scalable Macro-Pipelined FFT architecture for concurrent multi-symbol processing on SDR/MIMO OFDM systems [134], [252]. This reconfigurable FFT architecture can perform the FFT computations of multiple data streams with variable FFT lengths. The proposed FFT architecture can be used in multi-protocol SDR wireless sys-

tems to perform all the FFT calculations by supporting single or multiple MIMO channels for each of the different protocols (SDR mode), or in large MIMO-OFDM systems with increased number of data streams and support for various bandwidth configurations (MIMO mode). The fully scalable architecture is independent of the FFT length of each data stream and the number of multiple data streams and it can be easily configured to match the FFT computational requirements of any multi-protocol SDR or single-protocol MIMO OFDM wireless system.

In the rest of this chapter a configuration with 4 data streams and FFT lengths of 128 $\sim$ 2048-points is considered as an example implementation of the proposed architecture. The FFT processor can be used in a SDR system, which supports LTE with 2 MIMO channels (128 $\sim$ 2048-points FFTs) and IEEE 802.16e (WiMAX) with 2 MIMO streams (128 $\sim$ 2048 points FFTs), for all the FFT computations. In cases of MIMO-OFDM single-protocol systems the proposed FFT engine can perform multiple FFT computations of the same length (MIMO mode), while supporting variable bandwidth requirements. A IEEE 802.16e (WiMAX) system (128 $\sim$ 2048 points FFTs) with up to 4 MIMO channels can be handled by a single FFT module.

The architecture's processing core consists of a parallel structure involving multiple radix-2 butterfly processors and memory banks. Each processor can be reconfigured at run time to compute FFTs of 32, 64 or 128 complex points, while an interconnection network allows each butterfly processor to load (store) data from (to) the memory banks of other processors. The proposed architecture can process a single FFT frame with maximum length or multiple smaller frames in parallel, while sustaining the required throughput for multiple data streams. An optimized interprocessor communication scheme and a novel conflict-free in-place addressing technique, result in a low processing latency architecture with reduced hardware resources and power consumption.

4.3.1 The FFT Organization

The basis of the proposed design is a parallel and fully pipelined architecture capable of processing multiple variable-length OFDM symbols concurrently. The FFT engine computes decimation in time (DIT) FFT algorithms of variable length by using in-place technique with radix-2 factorization. It consists of several parallel butterfly processors and memory

banks while an interconnection network is used to group the processors in sets of 2, 4, 8 or 16. Each processor can be reconfigured at run time to compute FFTs of 32, 64 or 128 complex points. As mentioned before the proposed architecture is fully scalable and can be configured to support multiple data streams and/or larger FFT computations with increased processing elements, but the example implementation we consider in this chapter supports up to four data streams and $128 \sim 2048$-points FFT calculations.

The execution of the radix-2 FFT algorithm for 32/64/128 points computations requires 5, 6 or 7 stages, respectively. At these stages, each processor P_i updates two points per cycle and uses only two memory banks for storing all the intermediate results: its private bank $B_{i,0}$ and its auxiliary bank $B_{i,1}$. FFTs of length 128, 256, 512, etc are computed by allowing the processors to cooperate in groups and execute the remaining stages of the algorithm, in parallel. During these higher stages of the FFT computation, the processor P_i uses the interconnection network to access the auxiliary bank $B_{k,1}$ of another processor P_k. The indices i and k are defined by the computation flow of the FFT algorithm (the required butterfly calculation) and by a specific conflict-free in-place technique described in the following subsections. The application of this novel technique leads to the reduction of the interconnection network and to the minimization of the total computation cycles: only half of the memory banks need to be shared among the processors (i.e. the auxiliary banks) and, moreover, no conflicts stall the execution of the algorithm. The main FFT organization, for the case of 16 parallel radix-2 butterfly processors, which supports up to 4 multiple data streams with variable FFT lengths ($128 \sim 2048$), is shown in Fig. 4.1.

The proposed FFT processor can be used as a FFT accelerator engine in multi-protocol SDR/MIMO OFDM systems with multiple data streams, to perform all the required FFT computations. Each of the input streams can be configured at run-time for the FFT computation of variable length. For this operational mode the minimum memory requirements, for the FFT calculations, is N words, where N is the maximum FFT length supported. The architecture computes either one FFT frame of N-points, or two frames of $N/2$-point or four of $N/4$, etc points in parallel, with the minimum memory requirements. The support for multiple data streams is based on the parallel processing elements architecture, which accelerates the maximum length FFT computations for each data stream, while processes multiple smaller FFT lengths in parallel. Furthermore, the fully scalable architecture can be

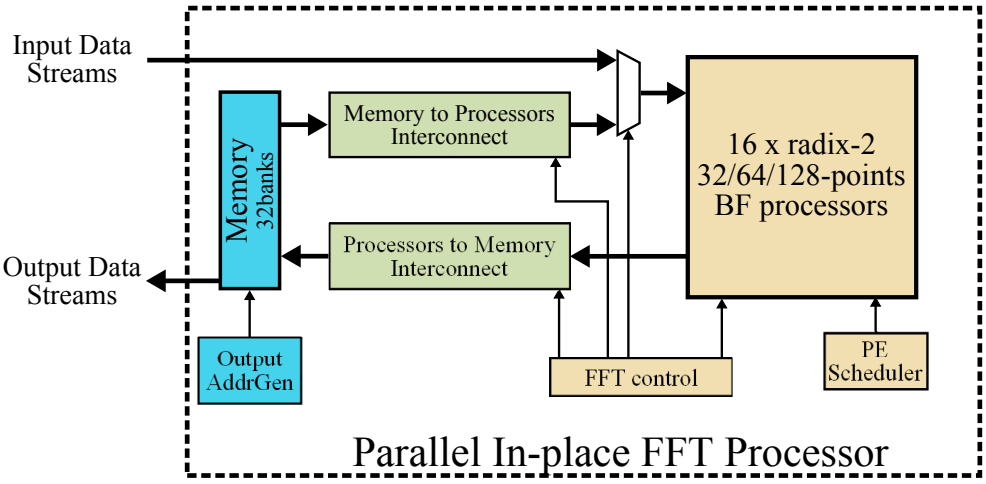


Figure 4.1 The main FFT organization

configured to support larger FFT sizes and/or increased number of multiple streams, with additional processing elements and increased memory requirements or it can configured to support smaller FFT sizes and/or decreased number of parallel data streams, with reduced processing elements and storage requirements.

The same FFT processor can be used in a multi-protocol SDR/MIMO pipeline path, as a continuous-flow FFT engine. For this operational mode the minimum memory requirements are $3 * N$ words per data stream, where N is the maximum FFT length supported. This FFT organization supports normal-order of parallel streams, for the input and output data. Simple re-ordering and bit-reversing circuits are required for the input data streams, while simple address generator modules are required for the output data. Additional simple mux-based interconnect circuits are used for the I/O buffers and computational memory selection. In the specific continuous-flow SDR/MIMO FFT architecture there is no requirements for delay memories/FIFOs, while the output data ordering is the same as the normal-order parallel input data streams. The overview of the continuous-flow SDR/MIMO FFT organization, for the case of up to four parallel data streams and maximum FFT length of 2048-points, is shown in Fig. 4.2. In the following subsections the case of a FFT organization which supports up to four multiple data streams of variable length up to 2048-points is considered.

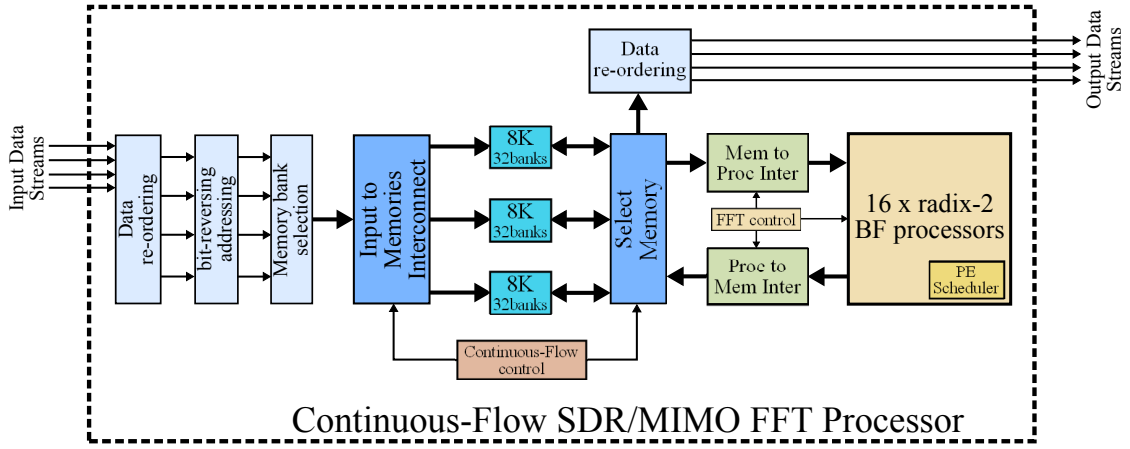


Figure 4.2 The overall continues-flow FFT organization

Processing Elements Scheduler

The reconfiguration and the grouping of the 16 radix-2 processors are determined by an internal scheduler, depending on the total number of parallel processing elements (PEs) and the length of the multiple FFTs (input FFT frames) at each time. This scheduler is responsible to efficiently group and configure the butterfly processors for multi-stream SDR/MIMO applications, with specific throughput and latency requirements. The scheduler design (for the case of four parallel data streams with variable FFT lengths $128 \sim 2048$) has focused on sustaining the required throughput rate of each data stream, while it has two distinct operational modes.

The first mode is used when there is at least one data stream with symbol length greater than or equal to 1024. For such cases, the scheduler will use all the processing elements, for the FFT computations of a specific data stream, while caching the input data from the other streams, for the next processing stage. The caching mechanism uses the input buffer to collect symbols of the same length, whose sum is 2048, i.e. 16 symbols of length 128, 8 symbols of length 256, etc. The collection of 2048 data belongs to the same data stream and constitutes the input to the 16-processor engine. The collections are processed sequentially in a round-robin fashion, sustaining an average throughput for each stream equal to its input rate. Each processor is configured to perform 128-points FFT.

The second operational mode handles the remaining cases (where all symbols have length $N \leq 512$). The scheduler assigns the symbol of each stream to a dedicated group

of four processors. The input streams are processed in parallel, as the processor groups operate independently from each another. Each processing element is configured to perform FFT with length equal to 1/4 of the symbol size assigned to its group, for maximizing the throughput while reducing the processing latency of the FFT computation per each data stream.

These two operational modes can handle both SDR and MIMO configurations, while a hybrid operational mode, based on these two, can be tailored to a specific SDR and/or MIMO system, for efficient and low area/power FFT calculations. Note that, in order to sustain the required throughput even in the worst case (4 FFTs of 2048 points each), the operating clock frequency of the FFT processor is set to $f_{op} = 1.375 \times f_{in}$, where f_{in} denotes the input sampling rate. Furthermore, the proposed FFT organization is independent from the FFT size of each of the four streams and it can be easy adapted to handle efficiently any SDR (multi-protocol), MIMO (single-protocol) or combinations of these two systems (e.g. multiple MIMO protocols).

Interconnection Network

To highlight the functionality, each processor P_i, $0 \leq i \leq 15$, performs radix-2 butterfly computations and has a memory of size 128 words divided into 2 banks ($B_{i,0}$, $B_{i,1}$). The architecture is designed to let each processor access two data words in parallel by realizing an *in-place* technique similar to [209] and complete each stage in 64, 32 or 16 cycles (128, 64 or 32-points FFT).

The interconnection network allows each processor P_i to access data from the memory bank $B_{x,1}$, of other four processors P_x, where x is the bitwise XOR of i $(= i_3 i_2 i_1 i_0)$ with 0001, $001\overline{i_1}$, $01\overline{i_2 i_2}$, $1\overline{i_3 i_3 i_3}$. For example, consider the 256-points FFT executed in P_{2k}, P_{2k+1}, $0 \leq k \leq 7$. Assuming that the processors are programmed (from the PE scheduler) to calculate 128-points FFT, the outcome of the 7th stage is 128 data with lower indices (the elements with initial addresses $d_0, \ldots, d_{127}$) stored in the two memory banks of P_{2k} with each bank storing a block of 64 elements. Furthermore, the 128 data with higher indices ($d_{128}, \ldots, d_{255}$) are stored in the banks of P_{2k+1} divided also into blocks of 64. The organization will have P_{2k} storing $d_0, \ldots, d_{63}$ in $B_{2k,0}$ and $d_{64}, \ldots, d_{127}$ in $B_{2k,1}$, while P_{2k+1} will store $d_{128}, \ldots, d_{191}$ in $B_{2k+1,1}$ and $d_{192}, \ldots, d_{255}$ in $B_{2k+1,0}$. This is accomplished by follow-

ing the permutations on the FFT data during the first 7 stages. The interconnection network allows P_{2k} to complete the upper half of the 8th stage by accessing the elements in $B_{2k,0}$ and $B_{2k+1,1}$, while P_{2k+1} completes the lower half of the 8th stage by accessing the elements in $B_{2k+1,0}$ and $B_{2k,1}$.

Data Flow

The data flow is similar to that in [209]. The in-place technique of [209] is modified to produce a sorted FFT output by using as a key the indices of the elements (the initial address of the elements). Initially all input elements are stored in banks $B_{i,0},\ B_{i,1}$ so that the LSB of each element's index specifies its storing bank. In the presented organization, the completion of each butterfly computation is followed by a permutation performed at the butterfly's output. To describe this permutation we consider the elements $x_s,\ x_r$ forming a transformation couple at stage (pass) j. The indices $x_s,\ x_r$ differ only at the jth bit and the results will be exchanging memory locations if the bit $(j+1)$ of the indices $x_s,\ x_r$ is 1. Also, an input permutation is performed: we will exchange $x_s,\ x_r$ at the butterfly input if x_s is stored in $B_{i,1}$. Proof of the technique can be found in [209] and the data flow is depicted in Fig. 4.3.

Fig. 4.3 shows the input and output permutations: the dotted lines denote that a transformation pair $x_s,\ x_r$ is loaded and stored without exchanging positions. Heavy red and blue lines denote the input and output permutations respectively. The figure also shows that the output elements are sorted with indices $0,\ldots,N/2-1$ stored in banks $B_{i,0}$ in increasing order and indices $N/2,\ldots,N-1$ stored in $B_{i,1}$ in decreasing order.

4.3.2 Butterfly Processor Architecture

The radix-2 butterfly processor P_i has two inputs (I_R, I_S) and two outputs (O_R, O_S), connected to the two dual-port memory banks $B_{i,0}$, $B_{i,1}$, the FFT control, the interconnection between the processor and the banks, the data address generation circuit and the twiddle factors address generation. Fig. 4.4 depicts the processor architecture.

Focusing on a 128-point processor as a reference case (the cases of 32- and 64-points are similar), the FFT control includes a 6 bits *up* counter to handle 64 pairs of data, at each computational stage and a *down* counter with 4 bits for the stages (passes). The interconnection

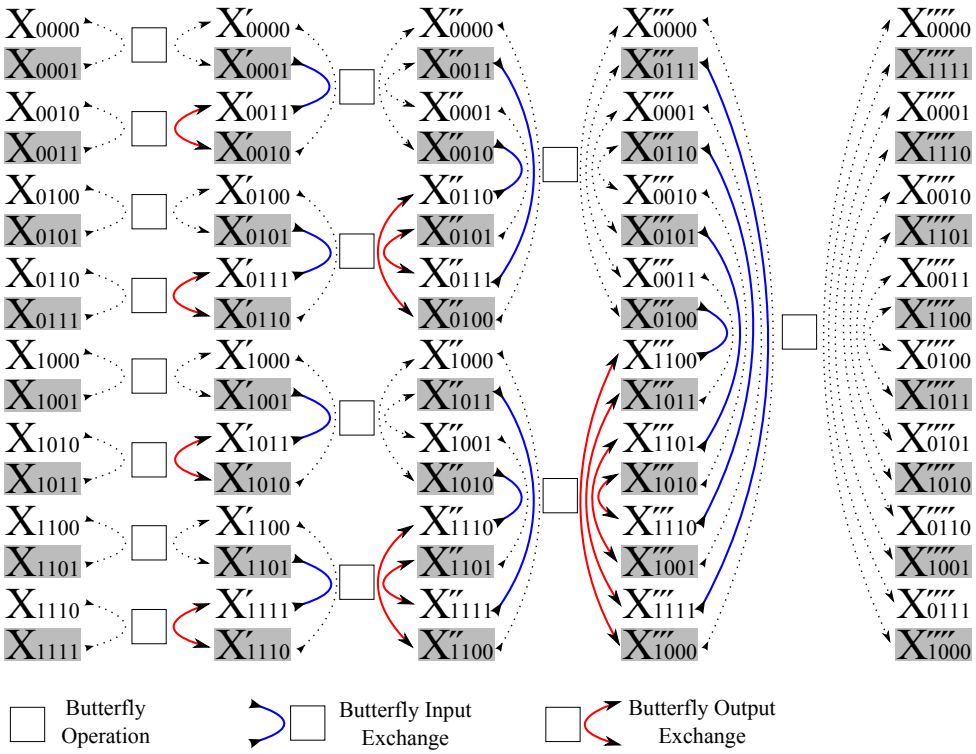


Figure 4.3 Data Flow of The FFT on 16 points. Unshaded elements are in $B_{i,0}$, shaded in $B_{i,1}$

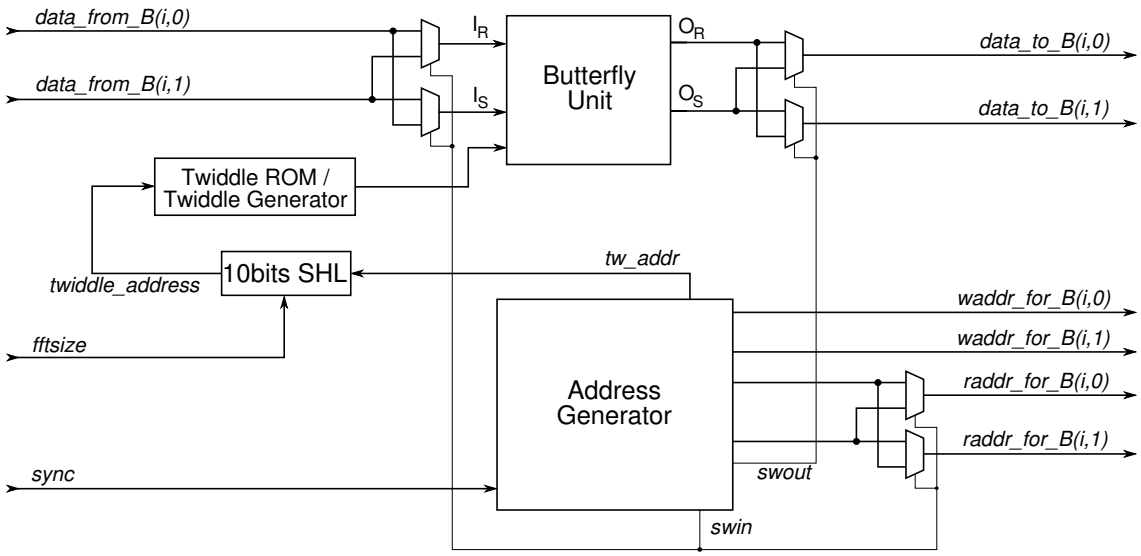


Figure 4.4 The radix-2 Processor

involves two multiplexers connecting the banks to each processor input and two multiplexers connecting the processor's outputs to each bank. These two sets of multiplexers are used to implement the input and output permutations of the proposed data flow. The radix-2 processor is organized to compute the FFT of 128 points by performing the Decimation-in-Time (DIT) algorithm and produces the 128 results sorted. After the FFT completion (7^{th} stage) the elements with indices $d_0, \ldots, d_{63}$ will be in the addresses $0, \ldots, 63$ of $B_{i,0}$ and the elements $d_{64}, \ldots, d_{127}$ will be in the addresses $63, \ldots, 0$ of $B_{i,1}$ respectively.

For the cases with more than seven computational stages ($N > 128$), each processor P_i is used for the calculation of an 128-point sub-FFT. Each of the stages include FFT computations in 64 pairs of data on each of the processors. For the first seven stages each butterfly unit accesses the data from its "local" memory banks, while for the stages $j > 6$ the interconnection network is responsible to connect one of the processor's input/output to the appropriate "auxiliary" memory bank of another processor. Each of the butterfly units performs the same operations, for the stages $j > 6$ on data stored in $B_{i,0}$, $B_{x,1}$ with x defined by the interconnection.

Data Address Generation

The data address generation circuit shown in Figure 4.5, uses the two control counters to generate the data addresses at each stage and the control signals of the 4 multiplexers. During the j^{th} stage the circuit will address $N/2$ pairs belonging to $N/2^{j+1}$ FFT sub-blocks. The circuit generates the addresses of the pairs by forming a word consisting of the 6 bits of the up counter. The addresses for $B_{i,0}$ ("BANK 0" in the Fig. 4.5) are generated by replacing the $(j-1)^{th}$ bit of the word format with a "0", while the addresses for $B_{i,1}$ ("BANK 1" in the Fig. 4.5) are generated by inverting the $j-1$ least significant bits of the $B_{i,0}$ address.

The circuit uses the 6 most significant bits of an 7-bit Left Rotator which has reset value "1111110b", to mark the $(j-1)^{th}$ bit of the *up* counter and a 6-bit AND-gate to construct the address for the $B_{i,0}$. The calculation of the $B_{i,1}$ address from the address of the $B_{i,0}$ includes a 6-bit Left Shifter (with reset value "0" and shift input value "1b") which marks the $j-1$ least significant bits of the $B_{i,0}$ address and a 6-bit XOR-gate to invert the marked bits and calculate the $B_{i,1}$ address.

The multiplexers at the butterfly outputs (O_R, O_S) realize the permutation and at each

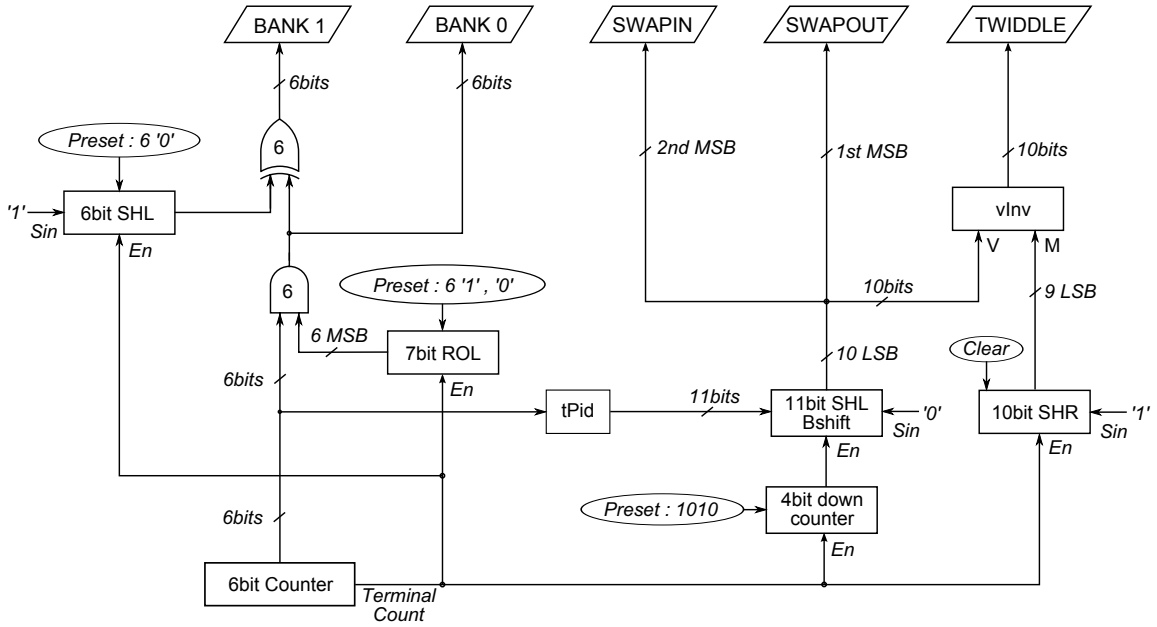


Figure 4.5 The Processor's Address Generator ("ROL" is a Left Rotator, "SHL" and "SHR" are Left and Right Shifter respectively)

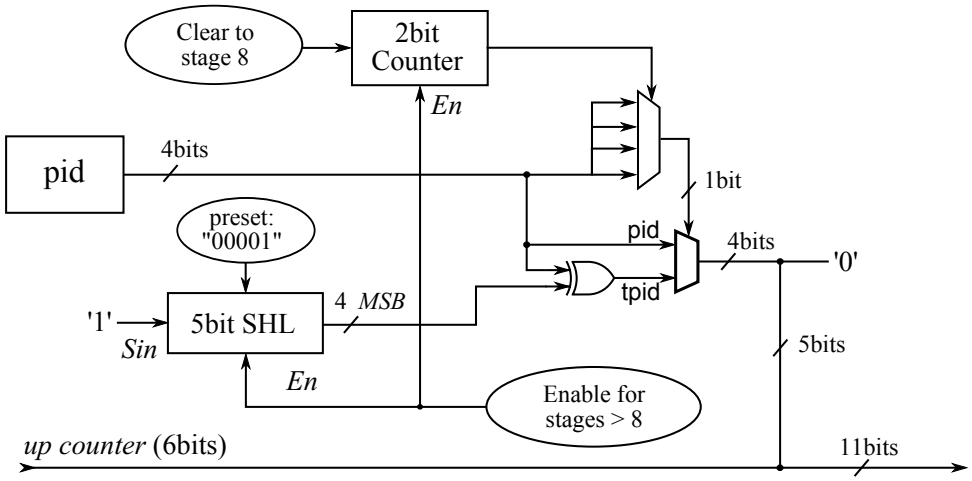


Figure 4.6 The Processor's Address Generator *tPid* module

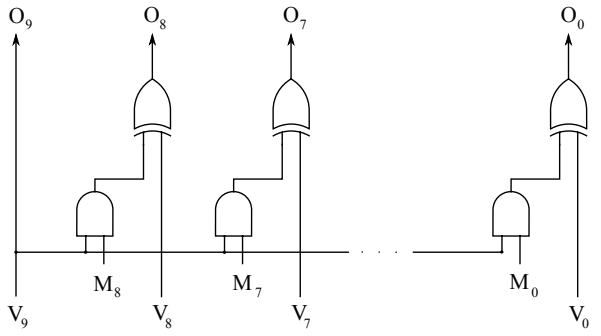


Figure 4.7 The Processor's Address Generator vInv module

pass j, $0 \leq j \leq 6$, are controlled by the j^{th} bit of the up counter: if the j^{th} bit is "1" then the multiplexers exchange the outputs of the butterfly ("*swapout*" signal of Fig. 4.5). The multiplexers at the I_R, I_S inputs of the butterfly are controlled by the $(j-1)^{th}$ bit of the up counter ("*swapin*" signal of Fig. 4.5). The multiplexers at the output of the address generator (read addresses) are also controlled by the "*swapin*" signal.

For the FFT stages $7 \leq j' \leq 10$ the calculation of the "*swapin*" and "*swapout*" signals depends on the processor index because a specific butterfly unit P_i should read (write) data from (to) the auxiliary memory bank $B_{x,1}$ of the processor P_x, where x is defined by the interconnection network. For these stages the module *tPid* of the address generator calculates the "*tpid*" value from the P_i ("*pid*") value and the stage number. Depending on the P_i value the *tPid* module selects the correct 5 bits prefix ("*pid*" or "*tpid*") for the 6 bits of the *up* counter. The module *tPid* is shown in Fig. 4.6.

Twiddle address generation

The larger FFT supported by the proposed architecture needs 1024 twiddle factors. The address generation circuits calculates the twiddle addresses (assuming that all the twiddles are stored in a ROM) at each step of the FFT computation. Before executing the last stage of a sub-FFT of size 2^{j+1} on the two sub-FFTs of size 2^j, the two sub-FFTs have their results sorted (Fig. 4.3) according to their indices. The results in the upper sub-FFT are sorted such that the 2^{j-1} lower indices are in increasing order in $B_{i,0}$ and the 2^{j-1} higher indices are in decreasing order in $B_{i,1}$. The lower sub-FFT are sorted such that the 2^{j-1} lower indices are in decreasing order in $B_{i,1}$ and the 2^{j-1} higher indices are in increasing order in $B_{i,0}$. Therefore, we read the twiddles of the first 2^{j-1} pairs by increasing a counter

and the twiddles of the remaining 2^{j-1} pairs by decreasing a counter.

This operation is simplified and accomplished as follows: At pass j we use the $j+1$ least significant bits of the *tPid* module output to create a 10 bit word: these $j+1$ bits are used as most significant bits (MSBs) followed by "0" (input V of the *vInv* module of Fig. 4.5). In case that the most significant bit (MSB) of the above word equal to "0" then the *vInv* module will forward its input as a twiddle address. In case that the MSB of the word equals to "1" then the *vInv* module will invert the remaining j MSBs (all the above $j+1$ MSBs apart from the MSB) to create the twiddle address. Fig. 4.7 shows the vInv module of the address generation circuit.

For the worse case FFT length we need 1024 twiddle factors stored in a ROM. We can reduce the memory requirements by taking advantage of the twiddle factor symmetries. As mention at Chapter 3 (eq. (3.12)) we can store only $N/8$ twiddle factors in ROM and calculate the rest $3N/8$ values. Based on (3.13) we can store only the values of $\sin\frac{2\pi k}{N}$ and $\cos\frac{2\pi k}{N}$ for $0 \leq k \leq N/8$ in two smaller ROMs. From these stored values we can calculate the A and B of (3.13). By using the symmetries of the twiddle factors we can reduce the memory requirements from 2048 real values (cos and sin values) to only 512. In the proposed organization each of the processors P_i has a dedicated twiddle ROM and by this technique we can reduce the ROM memory requirements by 24K words. An implementation comparison of this ROM memory optimization scheme and the implications of the aforementioned calculations (eq. (3.13)) in the fixed-point performance of the FFT processor, are presented in Section 4.4.

4.3.3 Interconnection Network

The 2048-point FFT transform is the largest size supported by the proposed organization. As mentioned above, the first 7 stages are completed by using only local memory addresses at processor P_i. In the worst case there will be four more stages (stages 7 to 10) to be computed and processor P_i will have to access the memories of at most 4 other processors. To simplify the analysis and the implementation, we fix P_i's upper input/output to always connect to $B_{i,0}$. Hence, the problem is reduced to assign to each P_i's lower input/output a set of banks, drawn from the superset of $B_{x,1}$ for all x. Furthermore, the set always contains the bank $B_{i,1}$ –which is used during the first 7 stages– and also contains four other banks, which

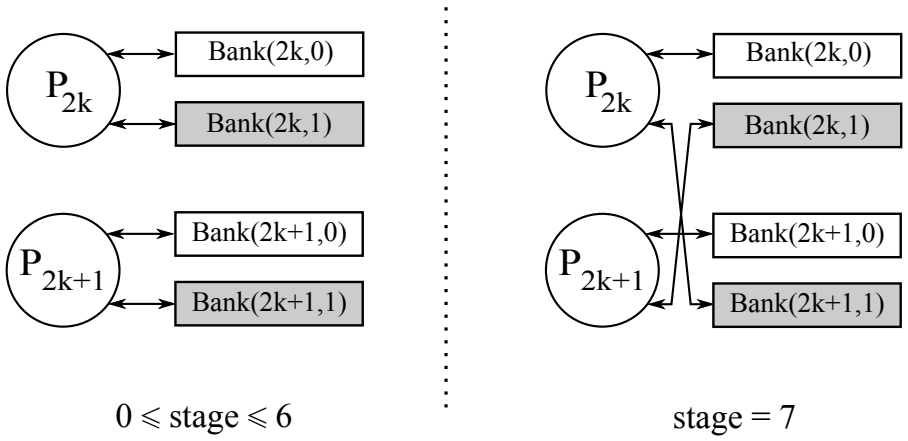


Figure 4.8 Processor and Memory Banks Interconnection (256-point FFT)

are utilized by (at most) four remaining stages.

The set of banks is determined by the flowgraph of the FFT algorithm as this has been mapped using the data flow of the in-place technique (Fig. 4.3). For the case of $N = 256$ points FFT it is straightforward to show from Fig. 4.3 that each processor P_{2k} must access data from the auxiliary memory bank of the "neighboring" processor P_{2k+1} only for the last stage (stage 7). Furthermore, each processor P_{2k+1} will access data from the auxiliary memory bank of the processor P_{2k}. Fig. 4.8 shows the processor and memory interconnection for the case of 256-point FFT calculation.

For the first seven stages of the calculation (stages 0 to 6), each processor (P_{2k}, P_{2k+1}) needs to access only each own memory banks (P_{2k} accesses $B_{2k,0}$ and $B_{2k,1}$, while P_{2k+1} accesses $B_{2k+1,0}$ and $B_{2k+1,1}$). For the last stage of the 256-point FFT computation (stage 7), the processor P_{2k} computes the first half of the transform, by accessing the $B_{2k,0}$ and $B_{2k+1,1}$ memory banks, while the P_{2k+1} processor computes the second half of the transform by using the $B_{2k+1,0}$ and $B_{2k,1}$ memory banks. Furthermore, according to the proposed data flow, the FFT results at the end of the 7^{th} stage can be retrieved by reading the $B_{2k,0}$ and $B_{2k+1,0}$ memory banks with increasing order, and $B_{2k,1}$ and $B_{2k+1,1}$ memory banks with decreasing order. No special addressing scheme is required by the proposed organization for the output data.

Assuming the worst case size of $N = 2048$ points, the data are initially divided into groups of 128 and assigned to processors in normal, increasing order (i.e. the first group to P_0, the second to P_1 etc.). To come up with the set of banks we restrict the analysis to

stages j after stage 6, and let $j = 6 + j'$, $1 \leq j' \leq 4$. We will use the binary representation of the index of the i^{th} processor P_i as: $i = [i_3 i_2 i_1 i_0]$. It is straightforward to show that by following the data flow of section 4.3.1, each processor P_i at the j^{th} stage will access data belonging to processor P_k: the index k (in binary $k = [k_3 k_2 k_1 k_0]$) is obtained by using i in the following two steps. First, we consider the effect of the output permutation which is a bitwise exclusive-or (XOR) operation on the index i with a 4 bit number containing j' ones in the j' LSBs and zeros otherwise. Second, data exchanges at the input occur at processors whose index has the $(j'-1)^{th}$ bit set. For these processors, the index k is computed by the first step calculation and corrected by performing another bitwise exclusive-or operation with a number containing $j'-1$ ones in the $j'-1$ LSBs (and zeros otherwise). Therefore, k is produced by super-imposing the two permutations (input and output) during stages $j \geq 8$ (stage 7 does not require correction).

More specifically:

$$\begin{aligned} k &= [k_3 k_2 k_1 k_0] \\ &= [i_3 i_2 i_1 i_0] \oplus [0 \ldots \underbrace{1 \ldots 1}_{j'}] \oplus [0 \ldots \underbrace{i_{j'-1} \ldots i_{j'-1}}_{j'-1}] \\ &= [i_3 i_2 i_1 i_0] \oplus [\, 0 \ldots 1 \underbrace{\overline{i_{j'-1}} \ldots \overline{i_{j'-1}}}_{j'-1} \,] \end{aligned}$$

Figure 4.9 shows the Processor and Memory Interconnection for the processor P_4, in the case of a 2048-point FFT calculation. At the first seven stages the processor accesses only its "local" memory banks $B_{4,0}$ and $B_{4,1}$. At the 7^{th} computational stage, processor P_4 accesses the auxiliary memory bank of P_5 the $B_{5,1}$, as in the case of 256-point FFT processor. In the following stages the P_4 accesses the auxiliary memory banks of the P_7, P_0 and P_{11} processors respectively.

Therefore, the interconnection network for each processor consists of a 5-to-1 multiplexer at the processor's lower input I_s and a 1-to-5 demultiplexer at its lower output O_s. The connections to each (de)multiplexer can be computed from the equation above. Note that the depth of the address calculation circuit, by using the proposed technique is constant, irrespective of the size N of the FFT transform, while it can be easily adapted to support larger FFT sizes and more radix-2 butterfly processors. Figure 4.10 shows the com-

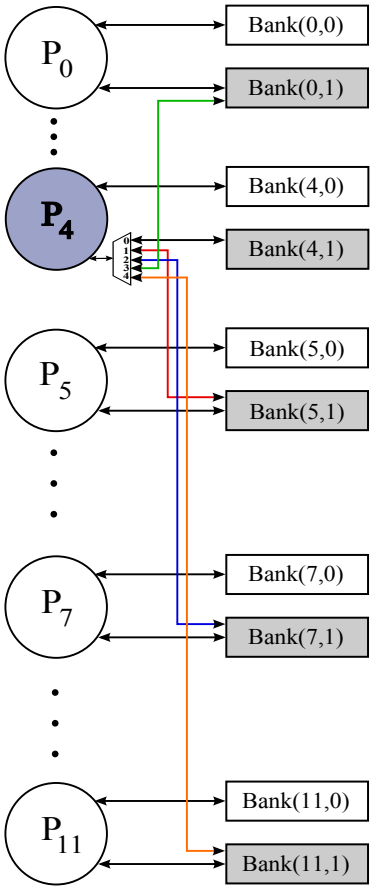


Figure 4.9 Processor and Memory Banks Interconnection (only for P_4)

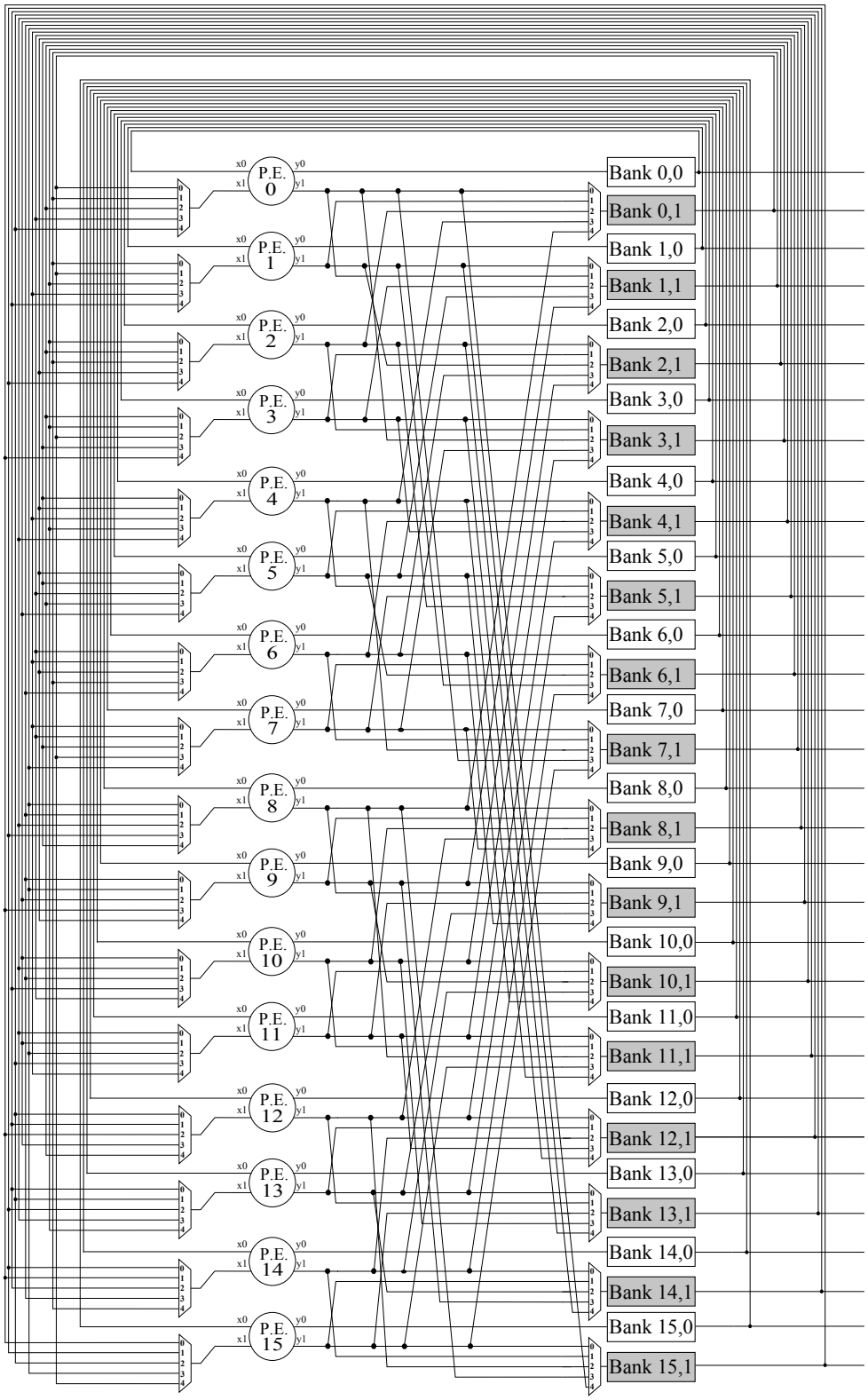


Figure 4.10 Processor and Memory Banks Interconnection

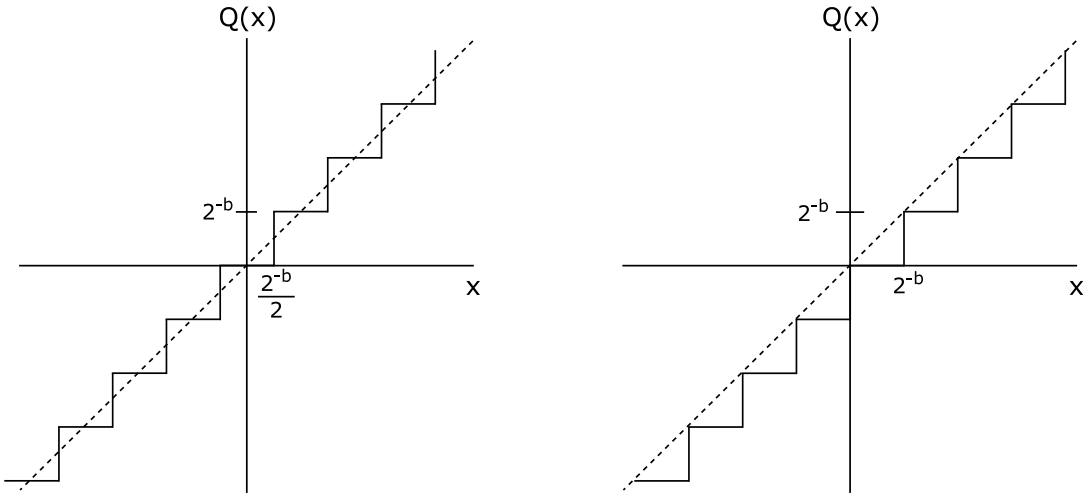
plete Processor and Memory Interconnection Network for the proposed FFT architecture, supporting up to 4 data streams and FFT sizes 128 $\sim$ 2048-points.

4.4 Fixed-point Error Analysis of the FFT

Digital signal processing applications require a finite register length to store intermediate results and coefficients in binary format. The effects of the finite word-length constraint appear in several different ways in applied DSP applications [222], [225], [322],[221],[331]. In the process of sampling a band-limited analog signal, the finite word-length constraint requires that the analog-to-digital conversion process produce only a finite number of possible values for each of the samples. The samples of the input must be quantized to fit a finite register length and this effect can be treated as an additive noise signal [220], [365], [27].

The results of the performed calculations over a finite word-length input values will naturally be numbers requiring additional bits for their representation. For example, a k-bit data sample multiplied by a k-bit coefficient results in a product that is $2k$-bits long. In cases of recursive DSP algorithms (filters, FFT, etc.) we should quantize the results of arithmetic operations, otherwise the number of bits will increase indefinitely. Similarly, in an FFT implementation the precise representation of the results of each stage requires $k+2$ bits more than that of the previous stage (assuming radix-2 butterfly and k-bit wide coefficients), due to the complex multiplication and complex addition operations inside the butterfly processor [220], [223], [182], [246].

In most of the FFT implementations for telecommunications all the calculations are performed with fixed-point rather than floating-point arithmetics [224], [222], [110]. Furthermore, it is natural to consider a register as representing a fixed-point fraction. In this way the product of two numbers remains a fraction and the limited register length can be maintained by truncating or rounding the least significant bits. These is no need for truncation or rounding the result of the addition of fixed-point fractions, with this type of representation. However, the magnitude of the resulting sum can exceed unity. This effect is commonly referred to as *overflow* and can be handled by requiring that the input data be sufficiently small so that the possibility of overflow is avoided. It is often useful to perform an approximate analysis by modeling the effect of truncation or rounding (quantizer) with an additive



(a) Rounding: $-\frac{1}{2}\cdot 2^{-b} < Q(x) - x \leq \frac{1}{2}\cdot 2^{-b}$ (b) Truncation: $-2^{-b} < Q(x) - x \leq 0$

Figure 4.11 Nonlinear relationships representing rounding and truncation.

error signal, which will be referred to as *roundoff noise*.

The most common methods for the FFT error analysis of the *Signal Quantization to Noise Ratio (SQNR)* and word-lengths are the statistical error analysis and the simulation-based analysis. The first method is based on statistical modeling the error sources (roundoff noise) and the results depend on the precision of the modeling [325], [220], [199], [222], while the second method compares the simulation results of the fixed-point computations with those obtained using the floating-point arithmetic [126], [237], [315].

4.4.1 Statistical Model for FFT Error Analysis

The analysis of the arithmetic roundoff of the FFT is based on a linear-noise model obtained by inserting an additive noise source at each point in the computation algorithm where roundoff occurs. To simplify the analysis we will emphasize on the Radix-2 Decimation in Time (DIT) FFT algorithm (Section 3.1) and we will make a number of assumptions.

Let us consider the representation of a signed number x in two's-complement with 1 sign bit and b_1 fractional bits ($|x| \leq 1$). The quantization of this number to b fractional bits introduces an error based on the method of the quantization (truncation or rounding). We

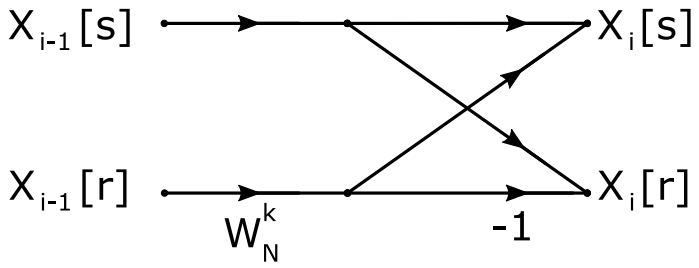


Figure 4.12 Radix-2 Decimation In Time Butterfly computation

can calculate these errors, assuming that $2^{-b_1} \ll 2^{-b}$ [223]:

$$\begin{aligned} &\textit{Truncation}: && -2^{-b} < E_T \leq 0 \\ &\textit{Rounding}: && -\tfrac{1}{2}\cdot 2^{-b} < E_R \leq \tfrac{1}{2}\cdot 2^{-b} \end{aligned} \tag{4.1}$$

Figures 4.11a and 4.11b show the input-output relation for two's-complement rounding and truncation, respectively.

As mentioned in Section 3.1 the N-point FFT is computed in $m = \log_2 N$ stages. The output array of each stage contains N complex numbers which are formed from the input array (output of the previous stage) by linear combinations of the elements, taken two at a time. For the radix-2 DIT algorithm, the basic 2-point FFT computation (butterfly) is of the form [222]:

$$\begin{aligned} X_i[s] &= X_{i-1}[s] + W_N^k \cdot X_{i-1}[r] \\ X_i[r] &= X_{i-1}[s] - W_N^k \cdot X_{i-1}[r] \end{aligned} \tag{4.2}$$

where the subscripts i and $(i-1)$ refer to the i-th and the $(i-1)$-th array, respectively, and s and r denote the location of the numbers in each array[1]. Figure 4.12 shows a flow graph representing the radix-2 DIT butterfly computation.

For the analysis of the roundoff noise effects we must associate an additive noise generator with each fixed-point multiplication. The butterfly of Fig. 4.12 must be replaced with the one of Fig. 4.13. The notation $Q[i,r]$ indicates the error resulting from quantization of the multiplication result of the $X_{i-1}[r]$ (r-th element of the $(i-1)$-th array) with the coefficient W_N^k. The input of the FFT and the coefficient W_N^k are in general complex numbers, so

[1]Note that $i = 0$ refers to the input array and $i = m$ refers to the output array

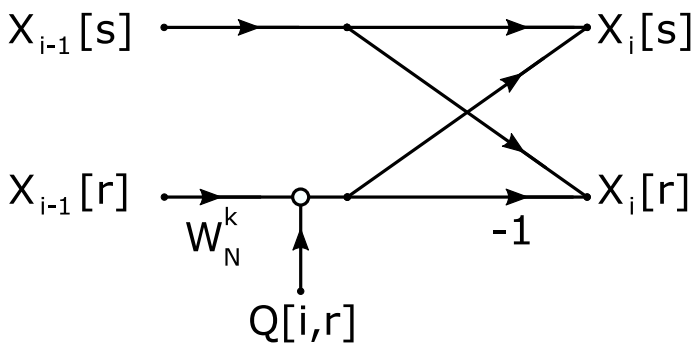


Figure 4.13 Linear-noise model for fixed-point roundoff noise in a Radix-2 Decimation In Time Butterfly computation

the multiplication in the Fig. 4.13 is complex and consists of four real multiplications[2]. We can assume the following properties for the errors of each of the real multiplications:

- The errors are uniformly distributed random variables over the range $-(1/2)\cdot 2^{-b}$ to $(1/2)\cdot 2^{-b}$, where the numbers are represented as $(b+1)$-bit signed fractions. Therefore, each error source has variance $2^{-2b}/12$.
- The errors are uncorrelated with one another.
- All the errors are uncorrelated with the input data and, consequently, also with the output.

Based on these properties we can evaluate the mean-square value for the roundoff noise of the complex multiplication:

$$\mathscr{E}\left\{|Q[i,r]|^2\right\} = 4\cdot\frac{2^{-2b}}{12} = \frac{1}{3}\cdot 2^{-2b} = \sigma_b^2 \tag{4.3}$$

From the flow graph of the decimation in time FFT in Fig. 4.14 we can observe that the transmission function from any node to any other node to which it is connected, is multiplication by a complex constant of unity magnitude (either unity or an integer power of W_N). Furthermore, each of the outputs of the N-point FFT is connected to (N-1) butterflies [222]. It is easy to evaluate the mean-square value of the output noise in the i-th FFT output

[2]There are implementations of the complex multiplication with three real multiplications but the accuracy of these implementations is less than that of the implementation with four real multiplications (see Section 4.4.2).

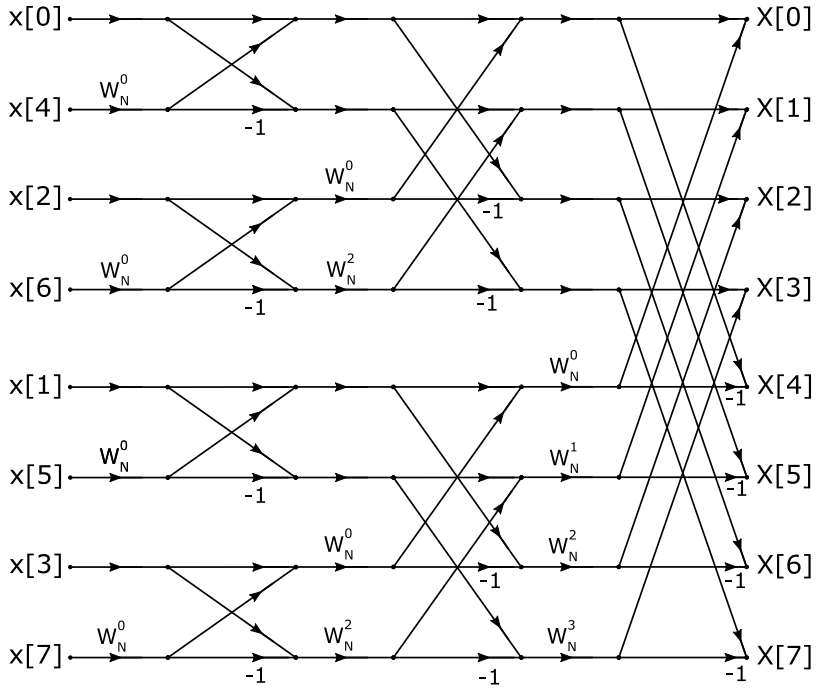


Figure 4.14 Flow graph of decimation-in-time decomposition of an 8-point FFT.

$F[k]$ and it is given by[3]:

$$\mathscr{E}\left\{|F[k]|^2\right\} = (N-1)\cdot\sigma_b^2 \simeq N\cdot\sigma_b^2 \tag{4.4}$$

Eq. (4.4) shows that the mean-square value of the output noise is proportional to N.

In the implementation of an FFT algorithm with fixed-point arithmetic we must guarantee no overflows [139] , [222], [315], [328], [322]. From (4.2) we can easily show that if the magnitude of the FFT output is less than unity then no overflow will occur in the calculations. This is equivalent with the magnitude of the input be less than $\frac{1}{N}$ ($|x[n]| < \frac{1}{N}$ for $0 \leq n \leq N-1$). In this case we can evaluate the average squared magnitude of the complex output of the FFT [222]:

$$\mathscr{E}\left\{|X[k]|^2\right\} = \frac{1}{3N} \tag{4.5}$$

From equations (4.4) and (4.5) we can evaluate the Signal Quantization to Noise Ratio

[3]The butterflies from first and second stages are not produce any roundoff errors due to the fact that the multiplication factor is unity or j. For simplicity we assume that all butterflies produce roundoff errors and we can consider the result as an upper bound on the roundoff noise.

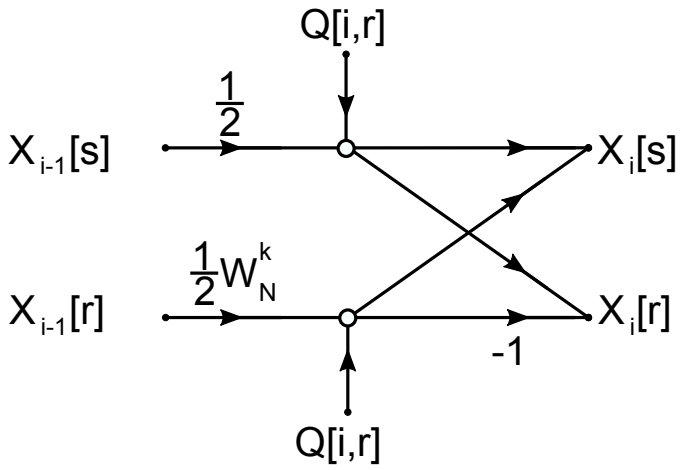


Figure 4.15 Decimation In Time Butterfly showing scaling multipliers and associated fixed-point round-off noise.

(SQNR):

$$SQNR = \frac{\mathscr{E}\left\{|F[k]|^2\right\}}{\mathscr{E}\left\{|X[k]|^2\right\}} = 3N^2 \cdot \sigma_b^2 = N^2 \cdot 2^{-2b} \tag{4.6}$$

This implies that the SQNR of the output of the FFT increases as N^2, or 1bit per stage.

From (4.2) we can also show that the maximum magnitude of the output of each stage increases by no more than a factor of 2. We can prevent overflow by requiring that the magnitude of the input of the FFT is less than unity ($|x[n]| < 1$ for $0 \leq n \leq N-1$) and by applying an attenuation of $\frac{1}{2}$ at the input of each stage. This alternative scaling scheme is shown in Fig. 4.15, where two noise sources are now associated with each butterfly. As in (4.3) we have:

$$\mathscr{E}\left\{|Q[i,r]|^2\right\} = \sigma_b^2 = \frac{1}{3} \cdot 2^{-2b} = \mathscr{E}\left\{|Q[i,s]|^2\right\} \tag{4.7}$$

The attenuation that each noise source experiences through the FFT flow graph depends on the stage in which is located. A noise source at the i-th stage of the FFT computation, will propagate to the output with multiplication by a complex constant with magnitude $(1/2)^{m-i-1}$, where $N = 2^m$. We can evaluate the mean-square magnitude of the roundoff noise at each output of the FFT:

$$\mathscr{E}\left\{|F[k]|^2\right\} = 4\sigma_b^2\left(1 - 0.5^m\right) = 4\sigma_b^2\left(1 - \frac{1}{N}\right) \tag{4.8}$$

For large values of N and based on (4.7):

$$\mathscr{E}\left\{|F[k]|^2\right\} \simeq 4\sigma_b^2 = \frac{4}{3}\cdot 2^{-2b} \tag{4.9}$$

From equations (4.9) and (4.5) we can evaluate the SQNR for the alternative scaling scheme on the FFT calculation:

$$SQNR = \frac{\mathscr{E}\left\{|F[k]|^2\right\}}{\mathscr{E}\left\{|X[k]|^2\right\}} = 12N\cdot\sigma_b^2 = 4N\cdot 2^{-2b} \tag{4.10}$$

This result is proportional to N rather than N^2 of the (4.6) and it is corresponding to half a bit per stage [325].

We should note that the dominant factor that causes the increase of the roundoff noise is the scaling of the signal magnitude required by the overflow constraint. From the analysis it is clear that the use of an $\frac{1}{2}$ attenuation at each stage of the FFT calculation is more efficient than an $\frac{1}{N}$ rescale of the input data in terms of SQNR performance.

4.4.2 Simulation-based Error Analysis

The simulation-based fixed-point error analysis is performed by comparing the output of a floating-point model, of the system under consideration, with the output of the fixed-point model of the same system, while the input data are the same for the two models. Figure 4.16 shows a block diagram of the simulation-based error analysis. The input data for the two models are quantized to the input word-length of the fixed-point model and the two outputs are compared by calculated the SQNR:

$$SQNR = 10\log_{10}\frac{\sum_{k=0}^{N-1}\left[X_{fl}(k)\right]^2}{\sum_{k=0}^{N-1}\left[X_{fl}(k)-X_{fp}(k)\right]^2} \tag{4.11}$$

where $X_{fl}(k)$ is the k-th output of the floating-point model, $X_{fp}(k)$ is the k-th output of the fixed-point model and N is the total number of output samples[4]. Furthermore, we can evaluate the fixed-point performance of a system by comparing the two outputs and calculate

[4]For more accurate results we must calculate SQNR for large number of output samples.

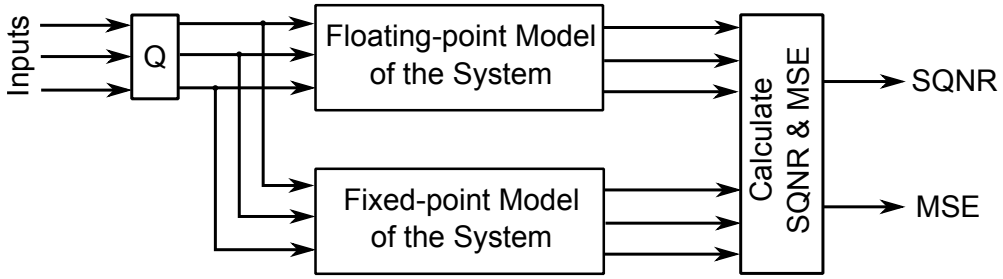


Figure 4.16 Simulation-Based Fixed-Point Error Analysis block diagram

the Mean-Squared-Error (MSE):

$$MSE = \frac{1}{N} \cdot \sum_{k=0}^{N-1} \left[X_{fl}(k) - X_{fp}(k) \right]^2 \tag{4.12}$$

In the following subsections we examine the simulation-based fixed-point error analysis for the module which performs the complex multiplication, the Radix-2 DIT butterfly and the proposed in-place FFT.

Complex Multiplication

We can perform multiplication of two complex numbers $z_1 = a + jb$ and $z_2 = c + jd$ with the following formula:

$$q = z_1 \cdot z_2 = (a + jb) \cdot (c + jd) = (a \cdot c - b \cdot d) + j(a \cdot d + b \cdot c) \tag{4.13}$$

This calculation requires four real multiplications and three real additions. There is an alternative computation [55] of the complex multiplication which requires only three real multiplications and five real additions:

$$\begin{aligned} m_1 &= (a+b) \cdot c \\ m_2 &= (d+c) \cdot b \\ m_3 &= (d-c) \cdot a \\ Re(q) &= m_1 - m_2 \\ Im(q) &= m_1 + m_3 \end{aligned} \tag{4.14}$$

In the case of the FFT computation one of the two operands in all the complex multiplications is a pre-computed twiddle factor (W_N^k). We can pre-compute and store all the intermediate results of (4.14) involving the real and imaginary parts of the twiddle factors to further reduce the number of real additions to three, in the calculation of $q = (a + jb) \cdot W_N^k$:

$$\begin{aligned} m_1 &= (a+b) \cdot Re\left(W_N^k\right) \\ m_2 &= f \cdot b \\ m_3 &= g \cdot a \\ Re(q) &= m_1 - m_2 \\ Im(q) &= m_1 + m_3 \\ f &= Im\left(W_N^k\right) + Re\left(W_N^k\right) \\ g &= Im\left(W_N^k\right) - Re\left(W_N^k\right) \end{aligned} \tag{4.15}$$

where the values of $Re\left(W_N^k\right)$, f and g can be pre-computed and stored in ROMs.

Comparing (4.13) and (4.15) we can see that the first implementation (complex multiplier form1) requires one real multiplier more, while the second implementation (complex multiplier form2), requires one real adder more and 50% more memory for storage of the twiddle factors. Furthermore, the two implementation of the complex multiplication have different fixed-point performance. To avoid overflow in the first implementation (4.13) we must scale the outputs of the real multipliers before the adders, while in the second implementation (4.15) we must scale the input of the complex multiplier (a and b), before any computation.

Figures 4.17 and 4.18 show the fixed-point performance of the two implementations in terms of SQNR and MSE, for several fixed-point modes (truncation (T) or rounding (R)) and several bit-widths (same for the two operants)[5]. Each of the plots contains the first implementation (CM form1) and two modes of the second implementation, one with truncation (CM form2 (T)) and one with rounding (CM form2 (R)) after the first real adder ($a+b$).

The first implementation of the complex multiplier (based on (4.13)) has better fixed-

[5]For the fixed-point simulations we use a uniform random distribution for the first operant and the twiddle factors of an 2048-point FFT for the second operant. The use of normal random distribution for the first operant has produce similar results. The SQNR and MSE values shown are the average of 10 iterations of 1M complex multiplication each.

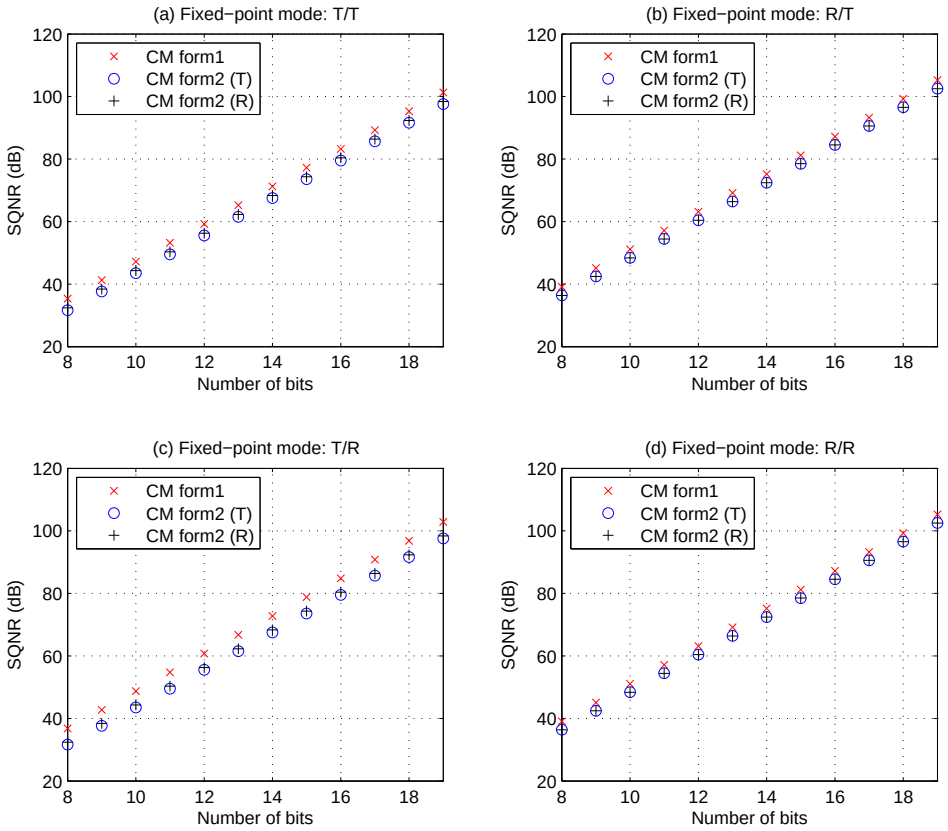


Figure 4.17 Complex Multiplier Implementations Fixed-Point Performance in SQNR for several bit-widths. In the form2 of the complex multiplier we can use either truncation (T) or rounding (R) after the first real adder. (a) Truncation (T) after real multipliers and real adders. (b) Rounding (R) after real multipliers and truncation (T) after real adders. (c) Truncation (T) after real multipliers and rounding (R) after real adders. (d) Rounding (R) after real multipliers and real adders.

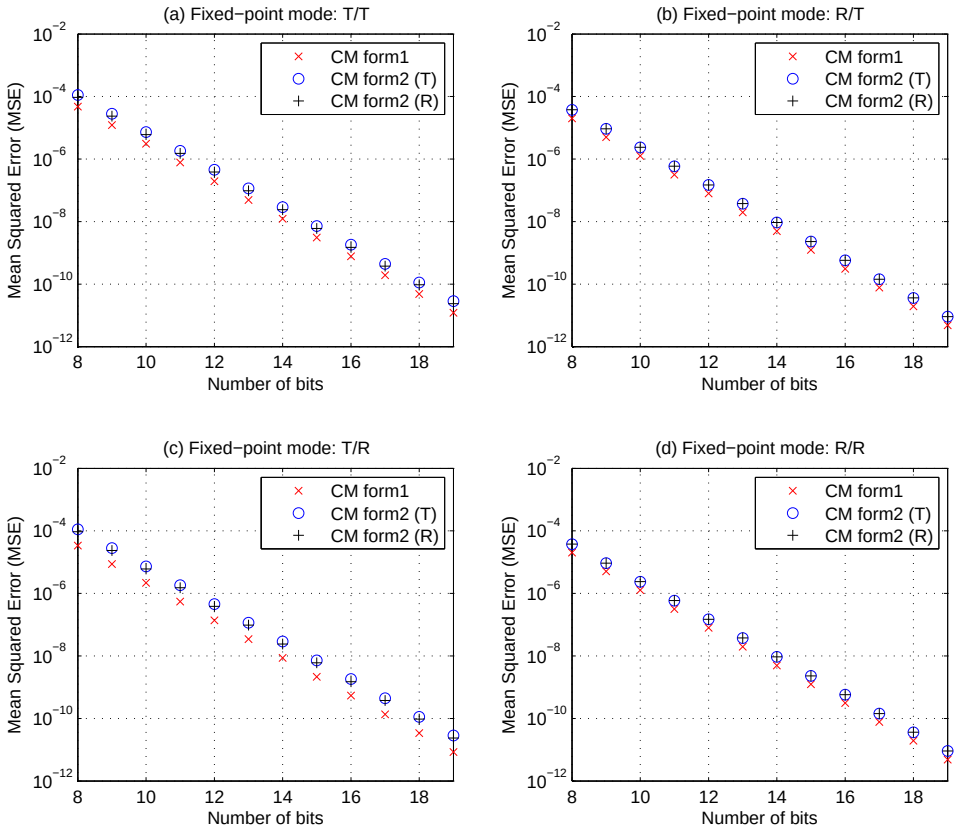


Figure 4.18 Complex Multiplier Implementations Fixed-Point Performance in MSE for several bit-widths. In the form2 of the complex multiplier we can use either truncation (T) or rounding (R) after the first real adder. (a) Truncation (T) after real multipliers and real adders. (b) Rounding (R) after real multipliers and truncation (T) after real adders. (c) Truncation (T) after real multipliers and rounding (R) after real adders. (d) Rounding (R) after real multipliers and real adders.

point performance for all bit-widths and all fixed-point modes. Fig. 4.17(c) shows that the second implementation (based on (4.15)) needs about one bit more for the same fixed-point performance (SQNR), in the specific fixed-point mode. Furthermore, we can see that for both implementations while the rounding at the output of the real multipliers performs about 4dB better than the truncation, the rounding at the output of the real adders has negligible performance increase compared to truncation. Finally, we can see from figures 4.17 and 4.18 that for the second implementation of the complex multiplier, the truncation (T) after the first real addition has similar fixed-point performance with rounding (R).

In the case of an FFT implementation it is crucial to minimize the datapath[6] bit-width for a required fixed-point performance. A wider datapath will result in wider memories for storing the intermediate results (output of each stage) and wider computational units in the butterfly module. We can increase the fixed-point performance of the FFT, for a specific datapath bit-width by increasing the bit-width of the twiddle factors. This will increase only the ROM in which the twiddle factors are stored and the complex multiplier unit inside the butterfly module.

Figures 4.19 and 4.20 show the SQNR and MSE performance of a complex multiplier with a fixed bit-width (10 bits for the real/imaginary part) for the first operant, which represents the datapath and a variable bit-width (from 8 bits to 19 bits for the real/imaginary part) for the second operant, which represents the twiddle factors. We can see from the plot in 4.19(a) that we can increase the fixed-point performance of the complex multiplier up to 7dB if the second operant is 3 bits wider than the first operant. Furthermore, we can see that the truncation at the output of the real multiplications has a small impact, in the fixed-point performance of the complex multiplier, compared to the rounding. When the two operants have fixed bit-widths (figure 4.17) the rounding at the output of the real multiplications performs about 4dB better than the transaction, while in the case of variable bit-width for the second operant (figure 4.19) the rounding performs only 1 or 2 dB better, when the bit-width of the second operant is 3 bits wider. Finally, we can see that if the bit-width of the second operant is more than 3 bits wider, than the bit-width of the first operant, there isn't any fixed-point performance boost at the output of the complex multiplier.

The technique of using wider bit-width for the second operant, of the complex multi-

[6] datapath is the path of data from one stage of the FFT to the next (Input/Output of the butterfly module).

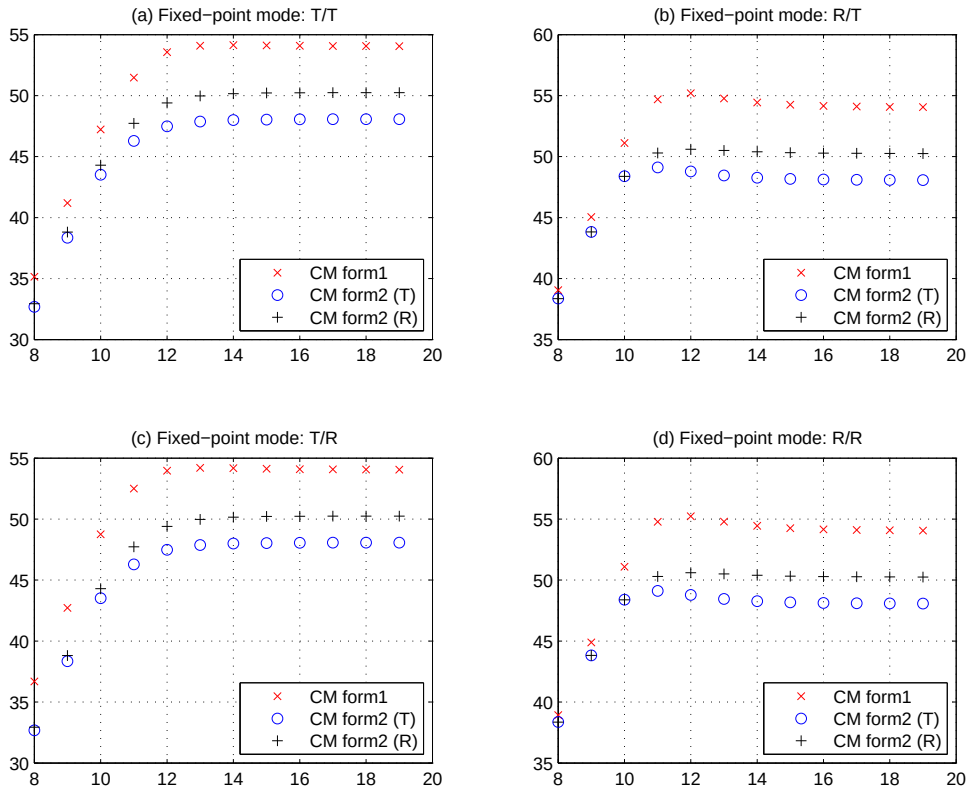


Figure 4.19 Complex Multiplier Implementations Fixed-Point Performance in SQNR: First operant (datapath) has a fixed bit-width of 10 bits while the second operant (twiddle factors) has a bit-width in the range 8-19 bits. The fixed-point modes are the same as in fig. 4.17.

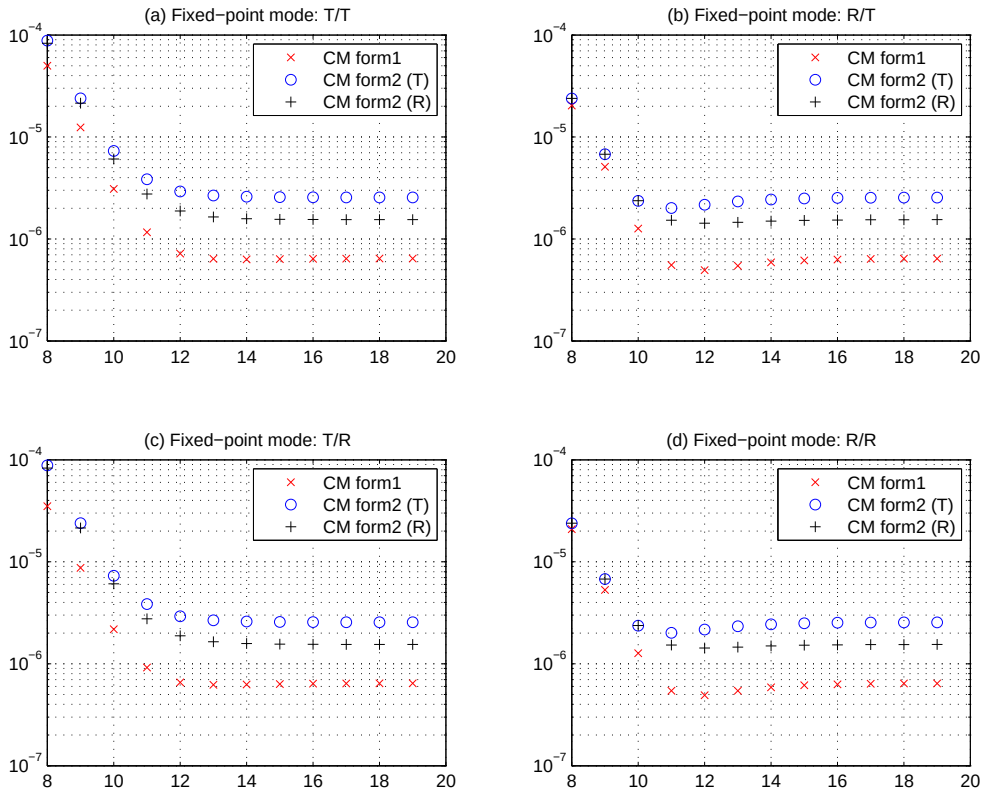


Figure 4.20 Complex Multiplier Implementations Fixed-Point Performance in MSE: First operant (datapath) has a fixed bit-width of 10 bits while the second operant (twiddle factors) has a bit-width in the range 8-19 bits. The fixed-point modes are the same as in fig. 4.18.

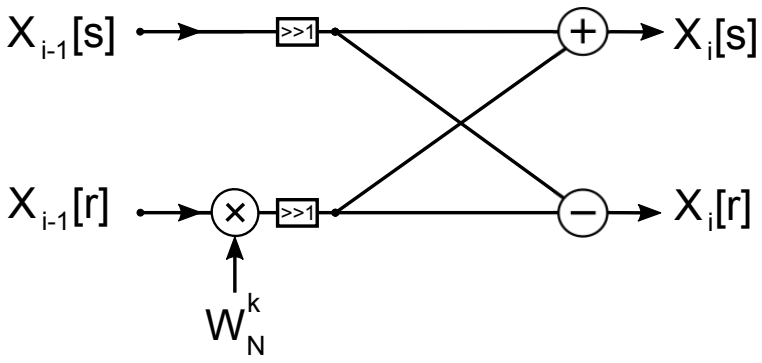


Figure 4.21 Radix-2 *Decimation in Time* butterfly unit.

plier, will increase the twiddle ROM requirements. We can use an optimization (based on equations (3.13)) to reduce the twiddle factors storage. Based on this optimization we can store only the first N/8 of the twiddles (for up to N-point FFT calculation) and calculate the rest of the factors at run-time. This modification will reduce the twiddle ROM for about 65-75% but it needs additional computations such as addition/subtraction of the twiddle factors and multiplication with a constant value. The fixed-point performance degradation of the FFT due to these computations is negligible but the extra calculations for the twiddle factors are in the critical path of the Radix-2 DIT butterfly unit.

Radix-2 DIT Butterfly Unit

Figure 4.21 shows a Radix-2 Decimation-in-Time butterfly unit. The upper output of the butterfly unit ($X_i[s]$) is the result of a complex addition between the complex multiplier result with the upper input of the butterfly ($X_{i-1}[s]$), while the other output of the butterfly ($X_i[r]$) is the result of a complex subtraction between the complex multiplier result, with the upper input of the butterfly unit ($X_{i-1}[s]$). Assuming that the real and imaginary parts of the two inputs of the butterfly are in the range of $[-1,1)$, same as the inputs of the FFT processor, then to avoid overflows, at the butterfly calculations, we should scale the data by a factor of $(1/2)$ [325], [139]. Figure 4.21 shows the right shift operations, for both butterfly inputs. As mention before one scale operation is needed inside the complex multiplier module, to avoid overflows at the addition/subtraction operation. The other shift operation is performed on the upper input of the butterfly unit, before the final add/subtract computation.

As with the complex multiplier, the scale operation can be performed with either trunca-

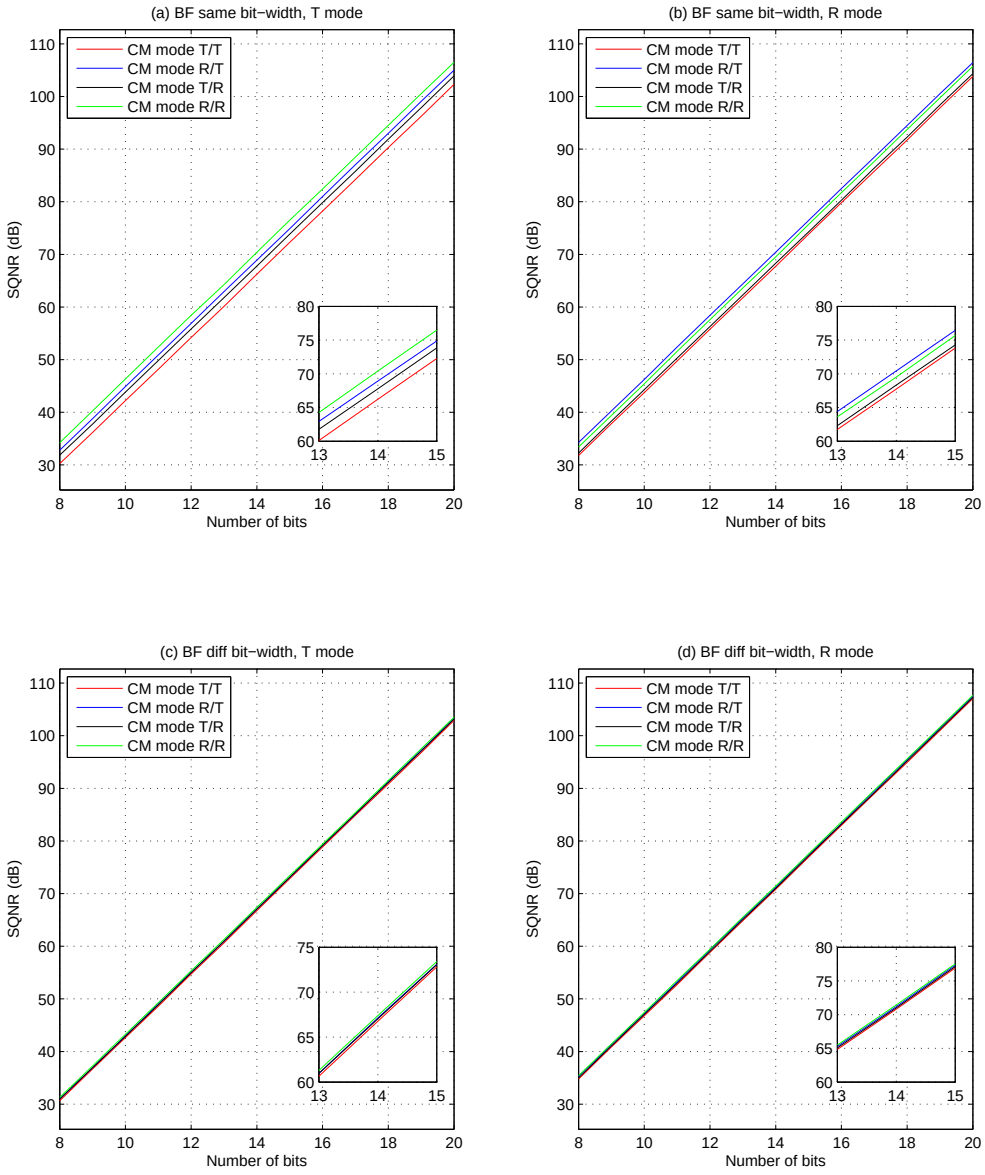


Figure 4.22 SQNR performance for the butterfly unit for several fixed-point modes: (a) The four CM form1 fixed-point modes, *same bit-width* for the BF inputs and twiddle factors and *truncation* on the BF outputs. (b) The four CM form1 fixed-point modes, *same bit-width* for the BF inputs and twiddle factors and *rounding* on the BF outputs. (c) The four CM form1 fixed-point modes, *different bit-width* for the BF inputs and twiddle factors and *truncation* on the BF outputs. (d) The four CM form1 fixed-point modes,*different bit-width* for the BF inputs and twiddle factors and *rounding* on the BF outputs.

tion or rounding of the results. Figures 4.22 and 4.23 show the SQNR and MSE performance of the Radix-2 DIT butterfly for several fixed-point modes. Only the complex multiplier of *form1* is used in the comparisons. Figures 4.22(a), 4.22(b), 4.23(a) and 4.23(b) show the SQNR and MSE performance of the butterfly when same bit-width is used for the input data and twiddle factors, for the four fixed-point modes of the complex multiplier and with truncation (a) or rounding (b) after the complex addition/subtraction of the butterfly calculations.

As expected the fixed-point mode with rounding after the real multiplications and after the real addition/subtraction (CM mode R/R in the figures) has better SQNR performance, when truncation is used for the butterfly outputs. In the case of rounding at the outputs of the butterfly, the fixed-point mode, of the complex multiplier, with rounding at the output of the real multipliers and truncation after the real addition/subtraction (CM mode R/T in the figures), has slightly better performance than the R/R mode.

Figures 4.22(c), 4.22(d), 4.23(c) and 4.23(d) show the SQNR and MSE performance of the butterfly when the bit-width of the twiddle factors is 3 bits wider than that of the input/output data. The four fixed-point modes of the complex multiplier has very limited performance improvement. The use of the truncation at the butterfly outputs results in loss of performance, compared to the use of the same bit-widths for the twiddle factors and the butterfly input/output data, for the case of R/R and R/T modes on complex multiplier.

The use of rounding operation, at the butterfly outputs, and different bit-widths for the twiddle factors and input/output data results in fixed-point performance boost. In the zoom box of figure 4.22(d) is shown a fixed-point performance of about 71-72dB for the butterfly, with input/output bit-widths of 14-bits and twiddle factors with 17-bits bit-width. The four fixed-point modes of the complex multiplier, has similar performance. In conclusion, the use of a 3-bit wider bit-width for the twiddle factors and rounding operation at the butterfly outputs, results in better fixed-point performance even with the use of truncation only complex multiplier (T/T mode in the figures).

In terms of hardware resources requirements the use of only truncation operations on the complex multiplier, while using wider bit-widths for the twiddle factors, reduces the number of adders with wide bit-widths (used for rounding). The rounding operations at the butterfly outputs is performed on smaller bit-widths and the resulting implementations has

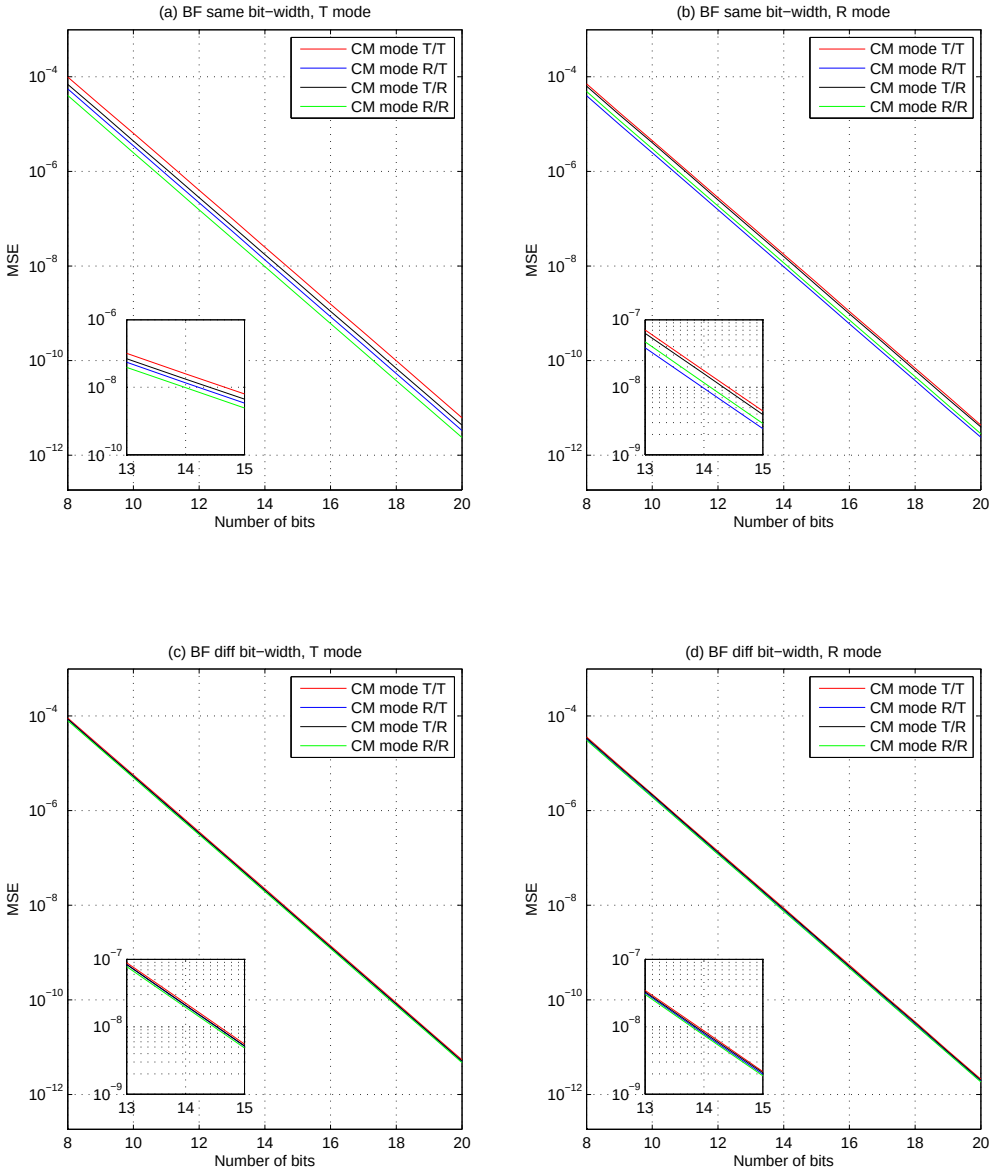


Figure 4.23 MSE performance for the butterfly unit for several fixed-point modes: (a) The four CM form1 fixed-point modes, *same bit-width* for the BF inputs and twiddle factors and *truncation* on the BF outputs. (b) The four CM form1 fixed-point modes, *same bit-width* for the BF inputs and twiddle factors and *rounding* on the BF outputs. (c) The four CM form1 fixed-point modes, *different bit-width* for the BF inputs and twiddle factors and *truncation* on the BF outputs. (d) The four CM form1 fixed-point modes,*different bit-width* for the BF inputs and twiddle factors and *rounding* on the BF outputs.

reduced hardware resources requirements. On the other hand, the wider bit-width of the twiddle factors results in increased storage requirements for the FFT processor hardware implementation.

FFT processor

The above analysis on the fixed-point performance of the complex multiplier and radix-2 butterfly units, for different rounding modes and bit-widths, reveals that for the implementation of the FFT processor, the most efficient selection, in terms of SQNR performance and hardware resources, is the form1 complex multiplier with truncation operations, the rounding at the outputs of the butterfly and the increased bit-width of the twiddle factors in comparison to that of the FFT data-path.

Table 4.1 $128 \sim 2048$ points FFT processor SQNR performance (dB) for different data-path and twiddle factors bit-widths

datapath bit-wdith	twiddles bit-width	FFT 128p	FFT 256p	FFT 512p	FFT 1024p	FFT 2048p
13	13	42.5775	39.9996	36.9824	33.8471	30.7593
13	14	45.8398	43.2858	40.3140	37.1574	34.0693
13	15	48.9832	46.6538	43.8144	40.7944	37.7589
13	16	50.5040	48.3334	45.5925	42.6597	39.6824
13	**17**	**51.1391**	**49.0417**	**46.3329**	**43.4228**	**40.4636**
13	18	51.3625	49.2860	46.6095	43.6857	40.7309
13	19	51.4714	49.3681	46.7014	43.7755	40.8161
14	14	48.6847	46.0533	43.0249	39.8688	36.7778
14	**15**	**52.1162**	**49.4743**	**46.4207**	**43.2462**	**40.1373**
14	16	55.2642	52.8551	49.9435	46.8934	43.8280
14	17	56.8167	54.5183	51.7500	48.7577	45.7504
14	18	57.3994	55.2196	52.4645	49.5050	56.5227
14	19	57.6093	55.4438	52.7028	49.7574	46.7859
15	**15**	**54.9246**	**52.2179**	**49.1290**	**45.9465**	**42.8324**
15	16	58.2780	55.5664	52.4893	49.3029	46.1709
15	17	61.4974	59.0050	56.0674	52.9593	49.8970
15	18	63.0534	60.6924	57.8768	54.8321	51.8157
15	19	63.6621	61.3727	58.5652	55.5779	52.5912

Table 4.1 and Figure 4.24 show the SQNR performance of the proposed FFT processor for sizes from 128 to 2048 complex points, with different bit-widths for the data-path and

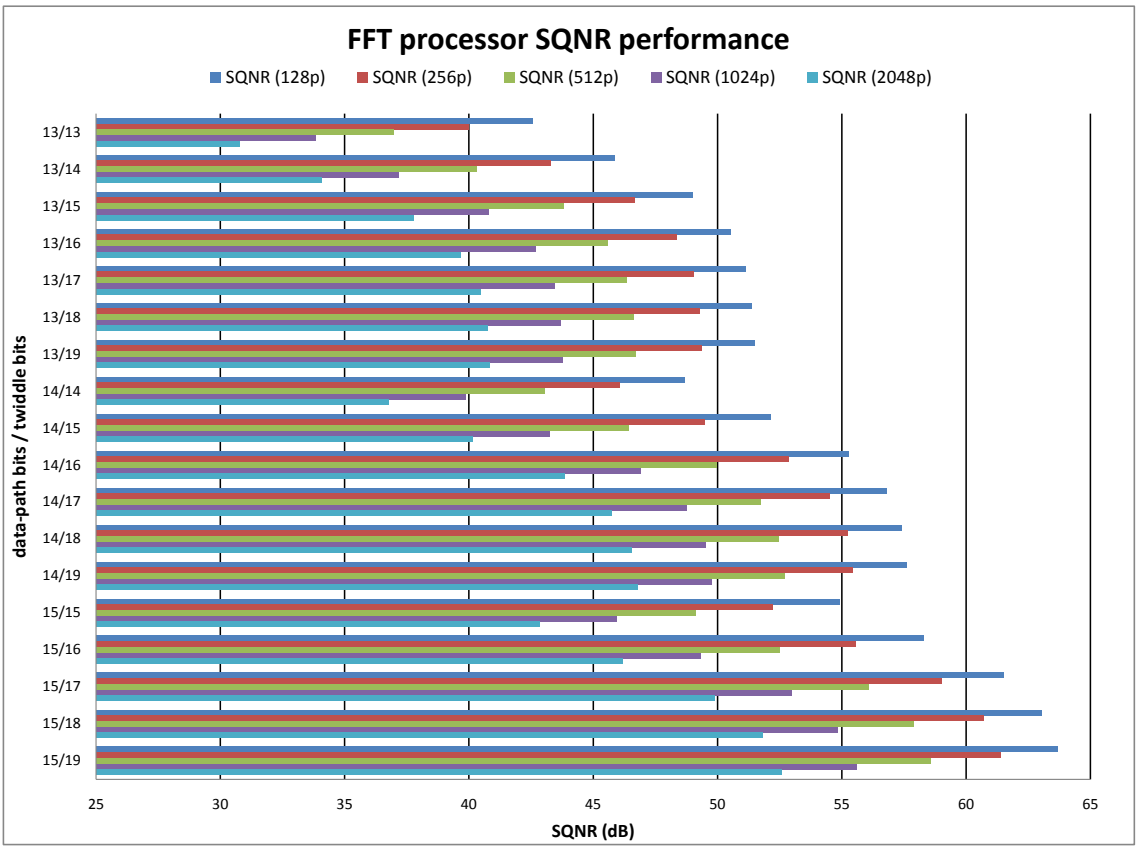


Figure 4.24 FFT processor SQNR performance for different data-path and twiddle bit-widths.

the twiddle factors. These results is based on a radix-2 butterfly with rounding operations at the outputs and the form1 complex multiplier with truncation operations. Furthermore, the SQNR simulations produce OFDM data for the input of the FFT processor, based on the 64-QAM modulation of the WiMAX (IEEE 802.16e) protocol[7]. The specific dense modulation is selected due to the fact that it is the most sensitive to noise, and the majority of the wireless protocols require an SQNR performance of about 38-40 dB, for the successful transmission of 64-QAM signals. These requirements are for the full transmitter processing path, which is not implemented in the SQNR simulations, but if we assume an ideal baseband processing path, then the SQNR performance of the fixed-point IFFT/FFT processor for all the supported FFT sizes should not be less than 40 dB, to meet the system requirements.

[7]The FFT frame is based on the OFDM structure of the WiMAX, in terms of null, pilots and data subcarrier indices.

From table 4.1 we can see that several configurations provide the required SQNR performance for the FFT processor. The three configurations, which result in SQNR performance above 40 dB for all the FFT sizes, with the minimum bit-width values for data-path and twiddle factors, are marked. The first configuration has the minimum data-path bit-width, among the three, but the wider bit-width for the FFT coefficients, which, most probably, result in increased hardware costs due to the high ROM memory requirements and wider complex multiplier units. The third marked configuration, in table 4.1 has the maximum data-path bit-width value and the same width for the twiddle factors, which is inefficient, in terms of SQNR performance, as mentioned in the previous subsections. Subsection 4.5 presents an analysis of the hardware costs for each of these three configurations.

Table 4.1 shows that for each FFT stage the SQNR performance is decreased by $\sim$3dB, which is about $\frac{1}{2}$ bit [325], if we consider that the 256-points FFT computation has one more stage than the 128-points FFT, etc. Furthermore, the use of 3-bits wider bit-width for the twiddle factors, in comparison with that of the data-path, results in an SQNR performance boost of about 7$\sim$8 dB, for all FFT sizes and all data-path bit-widths, shown in the table 4.1, as mentioned in the previous subsections. The SQNR performance boost, for configurations with more than 3-bits wider twiddles bit-width, is negligible (about 0.5dB). Finally, the configurations with 1-bit wider data-path and same "relative" twiddle factor bit-width (e.g. 13/13 with 14/14 and 15/15 or 13/16 with 14/17 and 15/18) have an SQNR performance boot of $\sim$6 dB.

Twiddle ROM optimization

As mentioned in Chapter 3 we can exploit the symmetry and periodicity of the FFT coefficients to reduce the ROM memory requirements of the FFT implementation. By using equation (3.11) a total amount of $N/4$ complex values can be stored in a ROM and a simple selection circuit with two adders can be used to produce all the $N/2$ twiddle factors. Further ROM memory requirements reduction can be achieved with the use of equation (3.12). A total number of $N/4+1$ real values can be stored in ROMs, while a more complex selection circuit can compute all the $N/2$ coefficients.

To minimize the computational latency of the FFT processor, the proposed architecture processes two complex FFT inputs per clock cycle, which results in a fully asynchronous

radix-2 butterfly. The address generation circuit (Section 4.3.2, Fig.4.5) computes the twiddle factor ROM address in parallel with the read addresses of the next FFT inputs and the write addresses for the current FFT outputs. The twiddle factor value should be available, at the radix-2 butterfly input, at the next cycle, in parallel with the two FFT inputs.

The synchronous ROM memory design[8] has a delay of one clock cycle, from the moment the read address is available at the address port until the specified data are available at the data port. For a full twiddle ROM implementations (1024 complex coefficient values in the ROM), the synchronous ROM design will not affect the critical path of the FFT processor, due to the fact that the needed coefficient value will be available from ROM at the next clock cycle, without any extra calculation. In cases of twiddle ROM optimization techniques, all the additional computations for the evaluation of the specific twiddle factor, from the output of the smaller ROMs, will increase the critical path of the radix-2 butterfly, and the FFT processor.

4.5 Implementation Results and Comparison

In this section we compare the proposed FFT architecture with a solution including four SDF radix-2 [298] FFT architectures, which also support variable symbol length (128 $\sim$ 2048 complex points FFT), in terms of hardware resource utilization. The implementation of the 4xSDF architecture can also support SDR (multi-protocol), MIMO (single-protocol) or combinations of these two systems (e.g. multiple MIMO protocols), as the proposed architecture. The comparisons include FPGA and ASIC implementations, with an FFT fixed-point performance of $\sim$40dB, with full twiddle ROM memory architecture, to minimize the effect of increased critical path due to extra twiddle computations. Synthesis trails (FPGA and ASIC) reveal that the most efficient configuration for the data-path and coefficient factors bit-widths, for an SQNR performance of 40dB, is the 14/15 scheme, in terms of hardware resources. For the FPGA implementations comparison, an average throughput of 24 MS/s per MIMO stream is considered (a total of 96MS/s), resulting on a clock frequency of 24MHz for the 4xSDF architecture and 34MHz for the proposed architecture, as

[8] In ASIC implementations large ROM memories are implemented with memory cells to minimize the occupied area. These ROM memory cells are synchronous (one clock cycle latency).

discussed in Section 4.3.1.

Each of the SDF FFT processor has 11 radix-2 stages to support the 2048-point FFT computation, while the radix-2 DIT butterfly of the first stage has only trivial multiplications and no complex multiplier unit is required. A total number of 40 complex multipliers are included in the 4xSDF architecture, while only 16 is required from the proposed architecture. Based on the analysis of Section 4.4.2 each of the complex multiplier units require four real multiplication modules. A fair FPGA implementation comparison will require all the real multiplication module to be implemented in the DSP blocks, to minimize the effects of uneven increased critical paths. A total number of 160 DSP blocks are required for the 4xSDF architecture, in comparison with the 64 DSP blocks for the proposed FFT implementation.

Table 4.2 FPGA implementation comparison

		Proposed Architecture	4xSDF Architecture	Gains (%)
XC4VFX140-11	(LUTs)	15569	23856	**34.74**
	(Slices)	9248	14333	**35.48**
	(DSP48E)	64	160	**60.00**
XC5VLX110T-2	(LUTs)	11401	14791	**22.92**
	(Slices)	2644	5303	**50.14**
	(DSP48E)	64	160	**60.00**
XC6VLX240T-2	(LUTs)	9441	12465	**24.26**
	(Slices)	2849	4483	**36.45**
	(DSP48E)	64	160	**60.00**

Table 4.2 shows the number of FPGA LUTs, Slices and DSP48E blocks occupied by the two MIMO FFT implementations, and the gains for the proposed architecture, for three different FPGA device families (Virtex-4, Virtex-5 and Virtex-6). All the FPGA devices include more than 160 DSP40E blocks, required by the 4xSDF architecture, for a fair comparison. The gains of the proposed architecture are more than 25% and 35% for the occupied LUTs and FPGA slices, respectively. Furthermore, due to limited requirements for DSP48E blocks the proposed architecture can be implemented in smaller/cheaper FPGA devices.

A reduction on the total number of complex multipliers, of the 4xSDF architecture, will require the use of more complex radix butterflies, such as radix-2^2 [95] and radix-2^3 [97]. The SDF architecture, with these butterfly structures, has reduced number of multiplication

modules, but more complex memory structures and increased complexity control units are required. Furthermore, for a variable length FFT processor with high-radix butterfly structures, there is the requirement for additional lower-radix butterflies, for the computation of all the FFT lengths.

In the ASIC implementation flow the synthesis tool performs several optimization iterations for both targeting frequency and for minimizing the occupied area. If the critical path of the design is relaxed, in the targeting frequency constraint, then more aggressive area optimization can be performed, otherwise less area can be saved. In the case of very strict clock constraints, in which the targeting frequency can not be achieved easily, aggressive speed optimizations should be performed by the synthesis tool, which can increase the area, due to register duplication, buffer insertion, etc.

Table 4.3 ASIC implementation comparison

Target Freq. (MHz)	Throughput (MS/s)	Proposed Arch. (KGates)	4xSDF Arch. (KGates)	Gain (%)
25MHz	100	127.77	236.72	**46.02**
50MHz	200	127.78	236.75	**46.03**
100MHz	400	127.77	237.25	**46.14**
200MHz	800	127.77	237.34	**46.17**
250MHz	1000	127.76	237.61	**46.23**
333MHz	1333	127.77	237.24	**46.14**
400MHz	1600	127.81	236.58	**45.97**
500MHz	2000	128.82	241.50	**46.66**
667MHz	2667	143.13	266.49	**46.29**

Table 4.3 shows the comparison of the proposed architecture and the 4xSDF implementation in terms of gate-count, for several target clock frequencies and total throughput. The proposed FFT processor has a gain of $\sim$46% in total KGates, in comparison to the solution with four distinct radix-2 SDF FFT processors, for all throughput configurations. The memory requirements (not included in the KGates numbers) for the proposed FFT processor is N words (7Kbytes), while the 4xSDF architecture requires $4N$ words (28Kbytes). For a continuous flow operation the 4xSDF architecture requires $4N$ additional memory for the bit-reversing, while the proposed architecture requires a total of $12N$ memory as mentioned in Section 4.3.1. Furthermore, the radix-2 SDF architecture includes small memories for

several computational stages, which can not implemented with memory cells with low-area and low-power performance. The proposed architecture includes several same size memory banks, which can be implemented as memory cells to save area and power consumption.

Compared to the 4x4 MR-MDC architecture of [133], which supports only 64 and 128-points FFT, the proposed architecture has almost double KGates and supports FFT computations up to 2048-points. As discussed in Section 4.4, more FFT computational stages require wider data-path and twiddle bit-widths, for the same fixed-point performance. Furthermore, the architecture of [133] requires extra memory structures to re-arrange the output data streams. The mixed-radix (radix-2/radix-4) MDC architecture of [349] has reduced butterfly and complex multiplication units, while having increased memory requirements for the re-arrange of the input data streams. The specific implementation can not be used in multi-protocol SDR systems, due to the restriction of the same FFT length for all the data streams, while the memory requirements for the re-arrange of the output data streams are not included in the comparisons. The complex mixed-radix scheme reduces the scalability of the architecture, to more complex MIMO-OFDM systems, with more data streams.

The butterfly and complex multiplier sharing scheme of [108], results in increased complexity memory structures for the re-arranging of the input data streams, with reduced scalability to more complex MIMO/SDR systems. The high-radix butterflies and the increased complexity of the control circuit have a negative impact in the total area and power consumption of the implementation. The 4x4 MIMO radix-4 MDC architecture of [352], has an efficient memory scheduling scheme to reduce the storage requirements for the input data streams shuffling, while the output sorting unit includes several large FIFOs and switching boxes to perform the data un-shuffling. The output data streams re-arranging is not including in the implementation, while the restriction of the same FFT size for all streams, reduce the usability of the architecture to multi-protocol SDR systems. The use of the MDC structure limits the scalability of the FFT processor to more complex MIMO systems.

The proposed radix-2 memory-based FFT architecture, for multi-protocol and/or MIMO-OFDM SDR systems, utilizes a reduced complexity address generation unit for the imple-

mentation of the in-place technique, to reduce the FFT processor latency and memory requirements. Furthermore, a simple interconnection network allows the 16 butterfly processors to communicate with data storage units, and perform parallel FFT computations of up to four variable length FFT symbols, ranging from 128 to 2048 complex points. The low-complexity FFT control and the butterfly scheduler unit reduce the processing latency by efficiently utilize the radix-2 processors, based on the FFT size of each of the data streams.

The reduced number of butterfly units and memory requirements of the proposed FFT architecture, is a key element for an efficient implementation of an SDR MIMO-OFDM system, in small FPGA devices. Furthermore, a gain of 46% in gate-count for the proposed architecture, in comparison with a solution with four distinct radix-2 SDF FFT processors can be achieved in ASIC implementations. The increased scalability of the proposed FFT processor and the ability to process different FFT sizes for each of the input data streams, is a unique characteristic, which make the propose architecture easily adaptable to any multi-protocol and/or multi-stream SDR OFDM system implementation.

Chapter 5

MIMO Detection

Wireless communication systems have a continuous growth in data-rates, performance, mobility and reliability. The increased bandwidth allocation and the high spectral efficiency of modulation and coding schemes help for this growth, but the key technology for the last decades is the use of Multiple-Input Multiple-Output (*MIMO*) systems. Wireless systems with 2,4 or 8 antennas are currently used for local area networks (802.11n/ac) or for cellular systems such as 3GPP-LTE and LTE-Advanced. Multiple antennas can bring diversity in order to enable more robust communication under tough link conditions, but more often they are used in order to increase the number of data streams within a given band, thanks to *Space Division Multiplexing (MIMO-SDM)*.

The use of space-division multiplexing results in a multi-path processing scheme in the transmitter, in which multiple data streams are transmitted in parallel on the different antennas. The detection of such signals have much more challenges. In a MIMO-SDM system with multiple transmit and receive antennas, the MIMO detector recovers the multiple transmitted data streams. Its task is to retrieve the symbol vector that was sent by the transmitter based on the received vector. The transmitted symbol vector consists of multiple complex symbols, each being one of the O possible points of a O-QAM constellation. In most of the cases, some inter-stream interference will be present and it may lead to strong degradations. This is especially the case when some streams are strongly attenuated by the channel or when multiple streams are not separable due to quasi-singular channel matrix.

5.1 Introduction

In MIMO systems with M_T transmit and M_R receive antennas, the MIMO detection problem can be summarized as the estimation of the transmitted symbols of size $M_T \times 1$, based on the received vectors of size $M_R \times 1$, with the assumption that the transmitted vectors have integer entries drawn from a finite alphabet O (O-QAM constellation) of size $|O|$. This problem is also known as *MIMO Demodulation*, *MIMO decoding*, *sequence estimation*, *closest point estimation*, *lattice decoding*, or *integer least-square (ILS) problem*.

The optimal solution, for the MIMO detection problem, is the *maximum likelihood (ML)* block detector, which is based on exhaustive search and has exponentially complexity $\mathcal{O}(|O|^{M_T})$, in term of the block size M_T (problem dimension). Reduced complexity, suboptimal decoders have been proposed by several researchers over the years. These can be classified into linear and nonlinear receivers. Linear receivers include zero-forcing (ZF) [280],[313] and minimum mean-square error (MMSE) detectors [31], [124], while non linear receivers include decision feedback (DF) [238], [160], [323], nulling-canceling (NC) [52], [75], [261], and variants relying on successive interference cancellation [159],[309]. These suboptimal detector schemes exhibits, in general, fixed complexity that is a polynomial function of the problem dimension, but may lead to considerable error performance degradation relative to ML decoding.

In the last years, researchers proposed realistic ML and near-ML detection algorithms, based on the idea of sphere decoding (SD). Exact ML or near-ML error performance can be achieved, with reasonably low memory requirements, for certain M_T, alphabet sizes $|M|$, and signal-to-noise ration (SNR) values, while the *average* decoding complexity is a polynomial function of M_T ($M_R \geq M_T$) [115], [311], [73]. Alternatively to sphere decoding algorithms (SDA), with only approximate ML performance but guaranteed complexity $\mathcal{O}({M_T}^4)$ for all SNR values is offered by the class of lattice reduction algorithms (LRA) [354], [332], [341], [340], [77]. Another approximate ML performance detection scheme is the probabilistic data association algorithm (PDA), which also has polynomial complexity [185], [235], [234], [149], [120]. Finally, semidefinite programming (SDP) algorithms have also been proposed for close to ML performance at complexity roughly $\mathcal{O}({M_T}^{3.5})$ [291], [187], [281], [292], [329], [203]. The sphere decoding algorithm (SDA) is the only one capable of returning the exact ML solution in general MIMO settings and appears to be more flex-

ible in terms of striking desirable error-complexity trade-offs, when it comes to near-ML performance.

5.2 MIMO System Model

In a MIMO-SDM system with M_T transmit and M_R receive antennas, the transmitted symbol vector $\mathbf{s}$ consists of M_T complex symbols, each being one of the $|O|$ possible points, when using O-QAM constellation. The received signal $\mathbf{y}$ is a vector of length $M_R \geq M_T$:

$$\mathbf{y} = \mathbf{H}\mathbf{s} + \mathbf{n} \tag{5.1}$$

where $\mathbf{H}$ is the $M_R \times M_T$ channel matrix and $\mathbf{n}$ is the noise vector of length M_R. Vector $\mathbf{n}$ is typically modeled as additive white Gaussian noise (AWGN) with zero mean and known variance $\sigma_n^2 \boldsymbol{I}$ (where $\boldsymbol{I}$ is the identity matrix), hence denoted as $\mathbf{n} \sim \mathcal{N}\left(\mathbf{0}, \sigma_n^2 \boldsymbol{I}\right)$. In wireless applications, $\mathbf{H}$ is random with entries drawn from a known distribution (e.g. Gaussian) and in typical cases, it adheres to a block fading model, according to which every realization of $\mathbf{H}$ remains invariant for a number of same consecutively transmitted vectors $\mathbf{s}$. Training symbols are used to estimate the channel over the blocks that it remains constant, which results in the assumption the $\mathbf{H}$ is known (estimated).

The traditional linear detectors, hard-output zero-force (ZF) and minimum mean-square estimation (MMSE), they are based on the following equations:

$$\hat{\mathbf{s}}_{ZF} = \mathbf{H}^{-1}\mathbf{y} = \left(\mathbf{H}^H\mathbf{H}\right)^{-1}\mathbf{H}^H\mathbf{y} \tag{5.2}$$

$$\hat{\mathbf{s}}_{MMSE} = \left(\mathbf{H}^H\mathbf{H} + \sigma_n^2\boldsymbol{I}\right)^{-1}\mathbf{H}^H\mathbf{y} \tag{5.3}$$

where $(...)^H$ denotes the conjugate (or Hermitian) transpose, and pseudo-inverse of $\mathbf{H}$ is $\mathbf{H}^{-1} = \left(\mathbf{H}^H\mathbf{H}\right)^{-1}\mathbf{H}^H$. The MMSE detector may be improved by an additional scaling step, based on (5.4) and (5.5). This scaling makes sure that the detected constellation is scaled back to the initial constellation and provides a small performance improvement. $\mathbf{S}$ is the

scaling matrix, out of which only the diagonal elements S_{ii} are needed:

$$\mathbf{S} = \left(\mathbf{H}^H\mathbf{H} + \sigma_n^2 \boldsymbol{I}\right)^{-1} \mathbf{H}^H\mathbf{H} \tag{5.4}$$

$$\hat{\mathbf{s}}_{i,Scaled} = \frac{\hat{\mathbf{s}}_{i,MMSE}}{S_{ii}} \tag{5.5}$$

At the other extreme, in terms of performance and complexity, the maximum likelihood (ML) detection considers all possible transmitted sets of symbols and computing a metric for each of them, at the output of the channel, and it can determine the most likely value for each of the transmitted symbols. The gap in error performance between a linear and the maximum likelihood detection comes from the fact that the ML detector exploits the full diversity of the channel, while linear detectors are limited to a diversity one (in case $M_T = M_R$). The maximum likelihood detector is designed to solve the following equation:

$$\hat{\mathbf{s}} = \arg \min_{\mathbf{s} \in O^{M_T}} \|\mathbf{y} - \mathbf{H}\mathbf{s}\|^2 \tag{5.6}$$

where O^{M_T} is the set containing all possible vector signals $\mathbf{s}$ of length M_T. Viewing $\mathbf{y}$ as a point in the M_R-dimensional space, the (5.6) suggests searching exhaustively over all possible $|O|^{M_T}$ candidate vectors $\mathbf{s}$ and selecting the one for which $\mathbf{Hs}$ (points of a lattice) lies closest to (has smallest Euclidean distance from) $\mathbf{y}$. This exponential complexity is prohibitive in most practical real-time scenarios where $|O|$ can typically range from 2 to 64 and M_T from 2 to 8. Sphere decoding algorithms can provide an ML or near-ML solution with polynomial complexity for a number of constellation sizes, SNR ranges and block sizes encountered in MIMO decoding applications.

5.3 Sphere Decoder Algorithm

As its name indicates, the sphere decoding algorithm solves (5.6) by searching exhaustively but efficiently all symbol candidates $\mathbf{Hs}$ (points of an M_R-dimensional lattice) within a hypersphere of radius r, centered at the received symbol $\mathbf{y}$ [73], [311]. The selection of the value of the hypersphere radius r and the efficient search over all the symbol candidates $\mathbf{Hs}$

inside the hypersphere, can reduce the exponential computational complexity (worst-case) of the ML detection, to polynomial (averaged over noise and lattice size) [28], [66], [93], [94], [115]. The two key questions for the sphere decoding algorithm is how can we choose the radius r of the hyper sphere and how we can determine which lattice points are inside the hypersphere.

A basic observation is that, although it is difficult to determine the lattice points inside a general m-dimensional sphere, it is trivial to do so in the one-dimensional case of $m = 1$. This is because the one-dimensional sphere reduces to the endpoints of an interval, and the desired lattice points will be the integer values that lie in this interval. We can use this observation to go from dimension k to dimension $k+1$. If we have determined all k-dimensional lattice points that lie in a sphere of radius r, then for any such k-dimensional point, the set of admissible values of the $(k+1)$-th dimensional coordinate that lie in the higher $(k+1)$ dimensional sphere of the same radius r forms an interval. This means that we can determine all lattice points in a hypersphere of dimension m and radius r by successively determining all lattice points in hyperspheres of lower dimensions (1,2,...,m) and the same radius r. An algorithm for the determination of the lattice points in an m-dimensional hypersphere, based on the above observation, essentially constructs a tree where the brances in the k-th level of the tree, correspond to the lattice points inside the hypersphere of radius r and dimension k. Moreover, the complexity of such an algorithm will depend on the size of three, or more precisely on the number of lattice points visited by the algorithm in different dimensions.

The lattice point $\mathbf{Hs}$ lies inside a hypersphere of radius r centered at $\mathbf{y}$ if and only if:

$$r^2 \geq d(\mathbf{s}) = \|\mathbf{y} - \mathbf{Hs}\|^2 \tag{5.7}$$

The QR decomposition [12] of the $M_R \times M_T$ matrix $\mathbf{H}$ is

$$\mathbf{H} = \mathbf{QR} \tag{5.8}$$

where $\mathbf{Q}$ is an $M_R \times M_T$ matrix with orthonormal columns ($\mathbf{Q}^H\mathbf{Q} = \boldsymbol{I}_{M_T}$) and $\mathbf{R}$ is an $M_T \times M_T$ upper triangular matrix. It can be shown [65] that

$$d(\mathbf{s}) = c + \|\hat{\mathbf{y}} - \mathbf{Rs}\|^2 \quad with \quad \hat{\mathbf{y}} = \mathbf{Q}^H\mathbf{y} = \mathbf{R}\hat{\mathbf{s}}_{ZF} \tag{5.9}$$

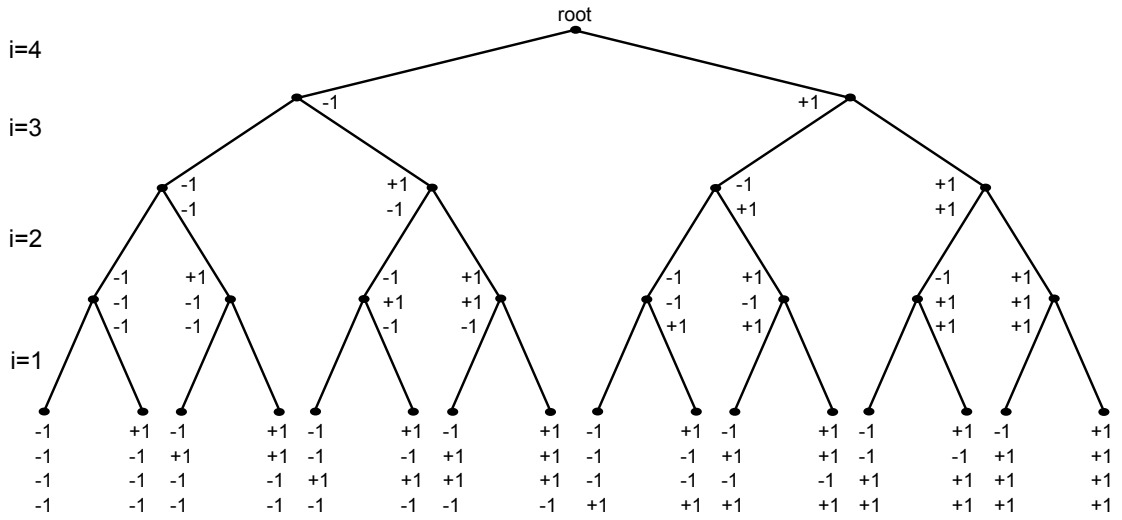


Figure 5.1 Search tree structure for a 4x4 BPSK MIMO system

where $\hat{\mathbf{s}}_{ZF}$ is the zero-force solution from (5.2). The constant c is independent of the vector symbol $\mathbf{s}$ and can be ignored (set $c = 0$). The QR factorization of the matrix $\mathbf{H}$ can be efficiently calculated with the use of several algorithms and techniques [111], [107], [186], [184], [229], [210].

We can build a tree in which the leaves at the bottom correspond to all possible vector symbols $\mathbf{s}$ and each level of the tree corresponds to a single transmit antenna, starting with antenna $i = M_T$ below the root node and ending with antenna $i = 1$ at the leaf nodes. Each of the nodes, except the leaves, has $|O|$ children, assuming an O-QAM constellation, and all nodes can be reached from *root* node by following the *branches*, which connect the nodes. Figure 5.1 shows the tree structure for the case of a 4x4 BPSK MIMO system. Each parent node has two children nodes, due to BPSK constellation, and starting from the root node we can reach all possible vector symbols $\mathbf{s}$.

In this tree structure, each node is a scalar symbol candidate $\mathbf{s}_i$. Therefore, any path from the root node to a node on level i corresponds to a *partial candidate symbol vector* $\mathbf{s}_i = [s_i, s_{i+1}, ..., s_{M_T}]^T$ with $\mathbf{s}_1 = \mathbf{s}$. The root node of the tree structure corresponds to an empty vector. Based on the above, we can recursively compute the (squared) distance $d(\mathbf{s})$ in (5.9) in a row-by-row fashion: Starting from the root node at level $i = M_T + 1$ and setting $D(\mathbf{s}_{M_T+1}) = 0$, the *partial (squared) Euclidean distances* (PEDs) $D(\mathbf{s}_i)$ can be evaluated:

$$D(\mathbf{s}_i) = D(\mathbf{s}_{i+1}) + \Delta(\mathbf{s}_i) \tag{5.10}$$

where $i = M_T, M_T - 1, ..., 1$ and $\Delta(\mathbf{s}_i)$ can be obtained as

$$\Delta(\mathbf{s}_i) = \left| \hat{y}_i - \sum_{j=i}^{M_T} R_{i,j} s_j \right|^2 . \tag{5.11}$$

Finally, $d(\mathbf{s})$ is the PED of the corresponding leaf: $d(\mathbf{s}) = D(\mathbf{s}_1)$.

Since the distance increments $\Delta(\mathbf{s}_i)$ are nonnegative, it follows that whenever the PED of a node violates the (partial) *sphere constraint* (SC) given by

$$D(\mathbf{s}_i) < r^2 \tag{5.12}$$

the PEDs of all its children will also violate the SC. Consequently, the sub-tree with root node this node can be pruned and the number of possible vector symbols (leaves of the tree) to be checked, can be reduced.

By the formulation of the MIMO detection problem as an optimization problem based on a weighted tree, various tree traversal approaches can be considered. The main two are the *depth-first* and *breadth-first* tree traversals. A common algorithmic complexity measure for a sphere decoding algorithm is the number of tree nodes, which the algorithm needs to examine their PEDs. These nodes are commonly named as *vising nodes*. The PED computations are the major calculations performed in the sphere decoding process and the number of these computations is generally a valid measure to compare different SD algorithms. Nevertheless, while this is the common case, specific SD algorithms may have low complexity, while vising more tree nodes. The initial value of the sphere constraint r^2 and the possibility of changing (updating) this value through the tree-traversal process can reduce the computational complexity of the sphere decoder, due to the tree pruning. Finally, the efficient "visiting" of the nodes, in a specific (*enumerated*) order, can further reduce the complexity, if we can safely eliminate nodes (or sub-trees) without checking their PEDs (without "visiting").

5.3.1 Tree Traversal Strategies

The worst-case computational complexity of the tree search problem for the MIMO detection is $\mathcal{O}\left(|O|^{M_T}\right)$, when an O-QAM constellation is used with M_T transmit antennas. This complexity is already a problem for a real-time 4x4 MIMO system with 16-QAM constellation and becomes more serious with higher modulation orders or more antennas. Various sphere decoding algorithms have been proposed in the literature in order to reduce both the average and the worst-case complexity, of the MIMO detection. Depending on the algorithm, a high complexity reduction is traded against a certain error performance (BER, FER, etc) reduction. The majority of these approaches are based on three optimization schemes: *Depth-First*, *Breadth-First* and *Best-First* tree traversals. The breadth-first algorithms mostly target a constant detection complexity and throughput, while the best-first and the depth-first approaches have variable complexity and throughput, based mainly on the channel condition.

Depth-First Sphere Decoding

Depth-first search approaches descend directly to a leaf by selecting the most promising branch on each level. The search continues on a tree level with further less promising branches until no branch fulfills the sphere constraint any more. In this case, the search returns to the next higher tree level and continues as long as the sphere constraint is fulfilled. This approach can be interpreted as descending into a local minima of $\Delta(\mathbf{s}_i)$ first and continuing with worse $\Delta(\mathbf{s}_i)$ on a tree level i, until the global minimum is found. In general this approach has variable runtime and throughput.

The depth-first traversal strategy imposes control and data flow dependencies which typically lead to sequential algorithm implementations. In cases of good channel conditions this scheme has low complexity and high efficiency, while the implementation needs to have sufficient and deterministic performance under worse channel conditions. Many algorithms, based on the depth-first tree traversal strategy, provide various approaches to impose constraints on the tree search for the worst-case scenarios. These constraints may have impact on the error performance of the sphere decoding, by providing a near-ML solution, while keeping the high throughput of the MIMO detection.

Schnorr and Euchner in [259] propose an optimization of the tree pruning, by defining a

search order for the child nodes on level i by the ascending metrics $\Delta(\mathbf{s}_i)$ of the candidates $\mathbf{s}_i$. The *Schnorr-Euchner (SE)* enumeration scheme has the additional complexity of finding and sorting the metrics $\Delta(\mathbf{s}_i)$ of all child nodes but also have the advantage of converging faster to the solution in the average-case scenarios, by fast pruning of the tree. Furthermore, an update of the sphere constraint r^2, each time a leaf node with $d(\mathbf{s}) = D(\mathbf{s}_1) < r^2$ is reached, is proposed in [259]. Therefore, the radius of the hypersphere is shrinking, while the algorithm is running, and significantly reduces the search complexity without sacrificing the ML optimality. The error performance and search complexity of the SD algorithm is independent from the selection of the initial sphere constraint, only if this constraint includes the ML solution. It is common to use infinite initial sphere constraint to ensure the ML solution. Several implementations are based on the Schnorr and Euchner enumeration scheme for reduced complexity maximum-likelihood MIMO detection on wireless systems [65], [29], [311], [271], [82].

Breadth-First Sphere Decoding

Breadth-first traversal strategy processes the tree level by level starting at the root and finishing at the leaves, instead of descending directly to a leaf as in the depth-first approach, targeting a fixed runtime and throughput. At each tree level, a certain subset of available branches are kept while others are discarded and no steps are made back towards the root node. The breadth-first algorithms need to take special care not to lose branches containing leaves close to the ML solution, otherwise a significant error performance degradation can be the cost for the advantage of fixed runtime and throughput. However, breadth-first algorithms are much better suited for parallelization than depth-first algorithms since the processing of a single tree layer includes several branches but much less dependencies. This is a significant advantage over depth-first searches particularly for low SNRs, but the complexity of breadth-first algorithms is not reduced or adapted at higher SNRs.

The most prominent breadth-first SD algorithm is the *K-best* sphere detection algorithm which initially proposed in [312]. These algorithms keep the best (most promising) K partial candidate symbol vectors $\mathbf{s}_i$, based on their PEDs, on every tree level i. Most of the K-best SD implementations have very low values of K, with low complexity for the required list-maintenance units, and close to ML error performance [327], [88], [44], [263].

Nevertheless, while K-best algorithms have low and constant algorithmic complexity, their error performance, most likely, depends on the channel conditions.

Another group of SD algorithms, which use the breadth-first traversal strategy, is the *Fixed-complexity Sphere Decoders (FSD)* initially proposed in [15] and [17], which are very similar to the K-best approach. The main difference of the FSD algorithms, compared to K-best, is the expansion strategy on each of the tree levels. In the K-best algorithms, for an O-QAM constellation and M_T transmit antennas, for each level i a (partial[1]) sorting of all the $|O|^{(M_T-i+1)}$ PEDs is required for the selection of the K best partial vectors $\mathbf{s}_i$, while the FSD strategy limits this process to the expansion of only n_i best children $\mathbf{s}_i$ for every parent node $\mathbf{s}_{i+1}$. Therefore, FSD has individual expansion degrees for every antenna and selects a sub-tree of $\sum_{i=1}^{M_T} n_i$ symbol vector candidates. Furthermore, a pre-processing step is required for the FSD algorithms very similar to a sorted QR decomposition calculation [15]. The error performance of the FSD is near-ML on average, while targeting a more regular control and data flow structure than the K-best leading to a reduced fixed complexity. Finally, trade-offs exists between the complexity and the performance of the FSD algorithms by manipulate the n_i parameters [16], [114], [113]. Very similar to the FSD approach is another breadth-first SD algorithm, the *Selective Spanning with Fast Enumeration (SSFE)* algorithm [156], [157], [218]. The SSFE employs also the fixed expansion technique with the difference of keeping (storing) the non-selected children nodes, while using a very efficient simplified enumeration scheme, based on geometrical heuristics approximations.

Best-First Sphere Decoding

A different approach in the tree traversal strategy is used by the best-first algorithms. This scheme establishes a list of partial candidate vectors which may be located in different sub-trees and on different tree levels and it was initially proposed in [207]. In each step, the search is continued with the partial vector with the lowest metric. The "tree-hopping" eliminates the disadvantage of the breadth-first algorithms to miss the ML solution. However, there is the risk not to reach any leaf while hopping across higher tree levels. Several modifications have been proposed to eliminate the disadvantages of the best-first tree traversal

[1]The majority of the K-best algorithms evaluate at most $K \cdot |O|$ PEDs, for each tree level, from which they select the K best partial vector symbols $\mathbf{s}_i$

scheme [39], [167], [268], with the depth-first like method to be the most efficient [167], [268]. The *Modified Best-First with Fast Descend (MBF-FD)* sphere decoder [167], which is a depth-first method modification on the best-first scheme, has a very efficient implementation supporting up to 8×8 MIMO systems.

5.4 Sphere Decoding with Soft Information

As mentioned before for a MIMO system with M_T transmit and $M_R \geq M_T$ receive antennas the transmitted coded bit-stream is mapped to a M_T-dimensional symbol vector $\mathbf{s} \in O^{M_T}$, where O is the set of complex-valued scalar constellation points (e.g. O-QAM constellation). Each symbol vector $\mathbf{s}$ is associated with a bit-level label vector $\mathbf{x}$, with entries $x_{j,b}$, where the indices j and b refer to the b-th bit in the label of the constellation point given by the j-th entry of $\mathbf{s} = [s_1\ s_2\ ...\ s_{M_T}]^T$. Soft information can be used by the channel decoder to increase the error-rate performance of the system. This will require a *Soft-Output* MIMO detection, which produce soft information in the form of *Log-Likelihood Ratios (LLRs - $L(\cdot)$)* for all bits in the label $\mathbf{x}$. For computational complexity reduction, it is common to employ the *max-log approximation* for the LLRs computation on the MIMO detection unit [105]. Based on (5.1) and assuming statistically independent and identically distributed bits, the LLR value of a bit $x_{j,b}$ (of the bit-level label vector $\mathbf{x}$) is:

$$L\left(x_{j,b}\right) = \min_{\mathbf{s} \in X_{j,b}^{(-1)}} \|\mathbf{y} - \mathbf{H}\mathbf{s}\|^2 - \min_{\mathbf{s} \in X_{j,b}^{(+1)}} \|\mathbf{y} - \mathbf{H}\mathbf{s}\|^2 \tag{5.13}$$

where $X_{j,b}^{(-1)}$ and $X_{j,b}^{(+1)}$ are the sets of symbol vectors $\mathbf{s}$ that have the b-th bit in the label of the j-th scalar symbol equal to -1 and +1, respectively. For each bit, one of the two minimum in (5.13) is given by the metric $\lambda^{ML} = \|\mathbf{y} - \mathbf{H}\mathbf{s}^{ML}\|^2$ associated with the ML solution of the MIMO detection problem of (5.6).

The other minimum in (5.13) can be written as:

$$\lambda_{j,b}^{\overline{ML}} = \min_{\mathbf{s} \in X_{j,b}^{\left(\overline{x_{j,b}^{ML}}\right)}} \|\mathbf{y} - \mathbf{H}\mathbf{s}\|^2 \tag{5.14}$$

where the *counter-hypothesis* $x_{j,b}^{\overline{ML}}$ denotes the binary complement of the b-th bit in the label of j-th entry of the $\mathbf{s}^{ML}$. Based on (5.6) and (5.14) the max-log LLRs can be written as:

$$L\left(x_{j,b}\right) = \begin{cases} \lambda^{ML} - \lambda_{j,b}^{\overline{ML}} & , \quad x_{j,b}^{ML} = -1 \\ \lambda_{j,b}^{\overline{ML}} - \lambda^{ML} & , \quad x_{j,b}^{ML} = +1 \end{cases} \tag{5.15}$$

From (5.15) we can conclude that an efficient maximum *a-posteriori* (MAP) MIMO detector should efficiently calculate $\mathbf{s}^{ML}$, λ^{ML} and $\lambda_{j,b}^{\overline{ML}}$ for $j = 1,2,...,M_T$ and $b = 1,2,...,Q$, where $Q = \log_2 |O|$.

After the QR decomposition of the channel matrix $\mathbf{H}$ we can transform the problem to a tree-search problem and use the sphere decoding algorithm for efficient calculation of the LLRs, according to the following equations:

$$\lambda^{ML} = \min_{\mathbf{s} \in O^{M_T}} \|\hat{\mathbf{y}} - \mathbf{R}\mathbf{s}\|^2 \tag{5.16}$$

$$\lambda_{j,b}^{\overline{ML}} = \min_{\mathbf{s} \in X_{j,b}^{\left(x_{j,b}^{ML}\right)}} \|\hat{\mathbf{y}} - \mathbf{R}\mathbf{s}\|^2 \tag{5.17}$$

We can use (5.10) and (5.11) to calculate the partial Euclidean distances (PEDs) of a partial symbol vector $\mathbf{s}_i$. Each path of the tree from the root node to a leaf corresponds to a symbol vector $\mathbf{s} \in O^{M_T}$. The solution of (5.16) and (5.17) corresponds to the leaf associated with the smallest metric in O^{M_T} and $X_{j,b}^{\left(\overline{x_{j,b}^{ML}}\right)}$ respectively.

The computation of the LLRs in (5.15) requires determining the metrics $\lambda_{j,b}^{\overline{ML}}$, which for given j, b is accomplished by traversing only those parts of the tree that have leaves in $X_{j,b}^{\left(\overline{x_{j,b}^{ML}}\right)}$. This calculation has to be performed for every bit and results in an order of magnitude increased computational complexity, compared to the hard-output sphere decoding. Considering the fact that by forcing the SD algorithm into specific subtrees, when computing the minimum of (5.17), leads to significantly less efficient tree pruning behavior, which results in an overall complexity increase of two orders of magnitude, compared to the hard-output sphere decoding case.

Tree Traversal Strategies for Soft-Output Sphere Decoding

In the case of depth-first tree traversal algorithms the *Repeated Tree Search (RTS)* and the *Single Tree Search (STS)* sphere decoding algorithms are support LLR computations. The main idea of the RTS algorithm [319] is to start by solving (5.16), by using the Schnorr-Euchner Sphere Decoding (SESD) algorithm (hard-output SD) and to then rerun the SESD to solve (5.17) for each bit in the symbol vector (QM_T times). A pre-pruned tree is used, when rerunning the SESD to determine the $\lambda_{j,b}^{\overline{ML}}$ in (5.17), and the decoder is forced to exclude all nodes from the search, for which $x_{j,b} = x_{j,b}^{ML}$. The radius update technique is used for all the SESD runs. The initial SESD run, which solves (5.16), have an initial infinite sphere constraint, while the next QM_T SESD runs, required to solve (5.17), have the initial sphere constraint equal to the minimum value of $\|\hat{\mathbf{y}} - \mathbf{Rs}\|$ over all $\mathbf{s} \in X_{j,b}^{\left(\overline{x_{j,b}^{ML}}\right)}$, found during the previous tree traversals, for complexity reduction without compromising the max-log optimality. The multiple hard-output SD runs of the RTS algorithm results in a large number of redundant computations, while the efficiency of the pruning behavior, when computing the $\lambda_{j,b}^{\overline{ML}}$, is very low. An increased pruning efficiency of the RTS algorithm is proposed in [193], by changing the detection order in each run, but the multiple QR decomposition steps increase the overall complexity of the decoder. A different expansion scheme is used in [270] to increase the efficiency for the *counter-hypothesis* SESD runs (solving (5.17)). Furthermore, a predefined and fixed complexity scheme is proposed, by using early termination of the decoder, to reduce the complexity and run-time of the LLR computation, by compromising the max-log optimality.

A more efficient tree traversal is used by the *Simple Tree Search (STS)* algorithm [116], [287], in which the search for the ML solution and all *counter-hypothesis* is performed concurrently. The update rules and the pruning criterion are based on a list containing the metric λ^{ML}, the corresponding label x^{ML} and the metrics $\lambda_{j,b}^{\overline{ML}}$. The main idea is to ensure that every node in the tree is visited at most once, by searching the subtree, originating from a given node, only if the result can lead to an update of at least one of the metrics in the list, either λ^{ML} or one of the $\lambda_{j,b}^{\overline{ML}}$. The list administration is based on two distinguished cases, when the decoder reaches a leaf. In the first case, when a new ML hypothesis is found, for each bit in the ML hypothesis that is changed in the process of the update, the metric of the former ML hypothesis becomes the metric of the new counter-hypothesis, followed by an

update of the ML hypothesis. This ensures that at all times $\lambda_{j,b}^{\overline{ML}}$ is the metric associated with a valid counter-hypothesis to the current ML hypothesis. For the second case, when $d(\mathbf{x}) \geq \lambda^{ML}$, only the counter-hypothesis have to be checked. For all j and b such that $x_{j,b} = \overline{x_{j,b}^{ML}}$ and $d(\mathbf{x}) < \lambda_{j,b}^{\overline{ML}}$ the decoder updates $\lambda_{j,b}^{\overline{ML}} \leftarrow d(\mathbf{x})$. The pruning criterion is compiled from two conditions to ensure that a given node and the entire subtree originating from that node are explored only if this could lead to an update of either λ^{ML} or of at least one of the $\lambda_{j,b}^{\overline{ML}}$ metrics. The STS algorithm outperforms the RTS strategy in terms of average complexity by a factor of 4 to 8 [287].

In the case of breadth-first tree traversal strategies there are extensions of the K-best algorithms to support LLR computations. The soft-output K-best sphere decoders require significantly larger values for K, than the hard-output variants, to ensure that all the counter-hypothesis metrics are included in the candidates list [44], [88]. For the cases that one or more counter-hypothesis metrics are not in the list, the LLR computation is performed based on the difference between the best and the worst metrics among the candidates list. These algorithms have fixed complexity and runtime, while their reduced error performance is close to optimal detection, on average. Furthermore, extensions on the fixed-complexity sphere decoders (FSD) have been proposed, for the soft-output LLR generation [18], [148], [338], [202]. These extensions are based on a similar idea than that for the K-best algorithm, to increase the size of the candidates list. Finally, several techniques and algorithms have been proposed to solve the problem of missing counter-hypothesis metrics in the candidates list in the breadth-first soft-output SD algorithms. The *Smart Ordering and Candidate Adding (SOCA)* technique [201], expands additional tree nodes considering always to include all the counter-hypothesis metrics for the current partial ML hypothesis. The *Parallel Candidate Adding (PCA)* scheme [336] is very similar to SOCA with different estimation of the partial ML hypothesis, for more regular implementation with reduced decoding latency.

Soft-Input Soft-Output Sphere Decoding

As mentions in Chapter 2 the *Soft-Input Soft-Output (SISO)* detection constitutes the basis for iterative decoding in OFDM and MIMO-OFDM systems, which, in general, achieves significantly better error-rate performance than decoding based on hard-output or soft-output detection algorithms. This performance gain comes at the cost of a significant and often pro-

hibitive in terms of practical implementation (mainly for MIMO-OFDM systems), increase in computational complexity.

The basic principle of the iterative detection and decoding, in a MIMO-OFDM system, is that the MIMO detector incorporate soft reliability information provided by the channel decoder, and the channel decoder incorporate soft information provided by the MIMO detector. These soft information exchanges between the MIMO detector and the channel decoder is performed in an iterative fashion until desired performance is achieved. The MIMO detector takes channel observations $\mathbf{y}$ and *a priori* knowledge L_1^A of the inner coded bits and computes new "extrinsic" information L_1^E for each of the $M_T Q$ coded bits per vector symbol $\mathbf{y}$ (assuming M_T transmit antennas and 2^Q-QAM constellation). The L_1^E is deinterleaved (if needed) to become the *a priori* input L_2^A of the SISO channel decoder which calculates extrinsic information L_2^E of the coded bits. Finally, the L_2^E is interleaved (if needed) and fed back as *a priori* knowledge L_1^A to the MIMO detector for the completion of one "iteration" [105].

Maximizing the *a posteriori* probability (*MAP* or *APP*) for a given bit minimizes the probability of error for that bit. The *a posteriori* probability is usually expressed as a log-likelihood ratio value (L-value), as mentioned above. Based on (5.1) the *intrinsic* LLRs of a SISO MIMO detector should be calculated according to [105]:

$$L_{j,b} = \log\left(\frac{P\left[x_{j,b} = +1|\mathbf{y},\mathbf{H}\right]}{P\left[x_{j,b} = -1|\mathbf{y},\mathbf{H}\right]}\right) \tag{5.18}$$

for all bits $j = 1,...,M_T,\ b = 1,...,Q$ in the label $\mathbf{x}$. An equivalent formulation is (Bayes's theorem):

$$L_{j,b} = \log\left(\sum_{\mathbf{s}\in X_{j,b}^{(+1)}} p(\mathbf{y}|\mathbf{s},\mathbf{H})P[\mathbf{s}]\right) - \log\left(\sum_{\mathbf{s}\in X_{j,b}^{(-1)}} p(\mathbf{y}|\mathbf{s},\mathbf{H})P[\mathbf{s}]\right) \tag{5.19}$$

where $X_{j,b}^{(+1)}$ and $X_{j,b}^{(-1)}$ are the sets of symbol vectors that have the bit corresponding to the indices j and b equal to $+1$ and -1, respectively. The prior $P[\mathbf{s}]$ is delivered by the channel

decoder in the form of *a priori* LLRs:

$$L_{j,b}^{A} = \log\left(\frac{P\left[xj,b=+1\right]}{P\left[xj,b=-1\right]}\right) \tag{5.20}$$

Based on the intrinsic LLRs in (5.19), the SISO MIMO detector calculates the *extrinsic* LLRs:

$$L_{j,b}^{E} = L_{j,b} - L_{j,b}^{A}, \;\; \forall j,b \tag{5.21}$$

that are forwarded to the channel decoder.

By using the max-log approximation and QR decomposition, as in the previous subsection, we can conclude that a SISO MIMO detector computes the intrinsic max-log LLRs according to

$$L_{j,b}^{D} = \min_{\mathbf{s}\in X_{j,b}^{(-1)}}\left[\frac{1}{\sigma_n^2}\|\hat{\mathbf{y}}-\mathbf{R}\mathbf{s}\|^2 - \log P[\mathbf{s}]\right] - \min_{\mathbf{s}\in X_{j,b}^{(+1)}}\left[\frac{1}{\sigma_n^2}\|\hat{\mathbf{y}}-\mathbf{R}\mathbf{s}\|^2 - \log P[\mathbf{s}]\right] \tag{5.22}$$

followed by the computation of the extrinsic max-log LLRs:

$$L_{j,b}^{E} = L_{j,b}^{D} - L_{j,b}^{A}, \;\; \forall j,b \tag{5.23}$$

For each bit, one of the two minima in (5.22) corresponds to

$$\lambda^{MAP} = \frac{1}{\sigma_n^2}\|\hat{\mathbf{y}}-\mathbf{R}\mathbf{s}^{MAP}\|^2 - \log P\left[\mathbf{s}^{MAP}\right] \tag{5.24}$$

associated with the MAP solution of the MIMO detection problem:

$$\mathbf{s}^{MAP} = \arg\min_{\mathbf{s}\in O^{M_T}}\left[\frac{1}{\sigma_n^2}\|\hat{\mathbf{y}}-\mathbf{R}\mathbf{s}\|^2 - \log P\left[\mathbf{s}\right]\right] \tag{5.25}$$

The other minimum in (5.22) can be written as

$$\lambda_{j,b}^{\overline{MAP}} = \min_{\mathbf{s}\in X_{j,b}^{\overline{MAP}}}\left[\frac{1}{\sigma_n^2}\|\hat{\mathbf{y}}-\mathbf{R}\mathbf{s}\|^2 - \log P\left[\mathbf{s}\right]\right] \tag{5.26}$$

where $X_{j,b}^{\overline{MAP}} = X_{j,b}^{\overline{x_{j,b}^{MAP}}}$ and $\overline{x_{j,b}^{MAP}}$ denotes the (bit-wise) *counter-hypothesis* to the MAP hypothesis. From (5.24) and (5.26) the intrinsic LLRs can be written in compact form as:

$$L_{j,b}^{D} = \begin{cases} \lambda^{MAP} - \lambda_{j,b}^{\overline{MAP}} & , \quad x_{j,b}^{MAP} = -1 \\ \lambda_{j,b}^{\overline{MAP}} - \lambda^{MAP} & , \quad x_{j,b}^{MAP} = +1 \end{cases} \tag{5.27}$$

Therefore, an efficient max-log-optimal SISO MIMO detection is reduced to efficiently identifying $\mathbf{s}^{MAP}$, λ^{MAP} and $\lambda_{j,b}^{\overline{MAP}}$, $\forall j,b$.

A SISO sphere decoding algorithm is more complex that the soft-output variant, due to the input soft information from the channel decoder (*a priori* LLRs). An approximation of the max-log solution in the SISO MIMO detection problem, have been proposed in [105], called *List Sphere Decoder (LSD)*. It is based on a depth-first search creating and maintaining a fixed-length list of candidate leaf nodes that are used as approximation for the sets $X_{j,b}^{(+1)}$ and $X_{j,b}^{(-1)}$. A vectorized version of the LSD algorithm have been proposed in [197], with some degree of parallelization for reduced decoding latency. For short list lengths the LSD algorithm has sufficient performance degradation and still variable complexity. An extension of LSD called tuple search has been proposed in order to overcome these issues [198],[265], [2].

An extension of the *Single Tree Search (STS)* algorithm towards an efficient max-log optimal SISO MIMO detection have been proposed in [286], [285],[25],[333]. To overcome the limitations of the SISO LSD algorithm, authors propose modifications in the list administration and tree pruning criterion to support the *a priori* LLRs. Furthermore, several complexity reduction schemes are introduced to reduce the increased computational requirements of the SISO MIMO detector.

An efficient complexity optimization for depth-first SISO SD algorithms is proposed in [216] and [166], which supports exact MAP performance. The introduction of a *Pruning Metric*, which is the value that it will be compared to the sphere constraint, for the tree pruning and an efficient handling of this metric, can result in increased number of "visiting nodes" but with decreased overall computational complexity. The proposed algorithm can be used with efficient enumeration schemes (Chapter 6) in a SISO depth-first search sphere detector, resulting in reduced computational requirements for the nodes enumeration procedure, performed in each visiting node.

Finally, some modification on breadth-first search SD algorithms have been proposed to support SISO [47], [14], [57]. These modifications are mainly on the FSD algorithm

and are based on changes in the administration of the list of candidates to reduce the risk of missing counter-hypothesis metrics. The increased parallelization supported by FSD algorithms results in reduced decoding latency for the SISO MIMO detector, with significant performance degradation.

5.5 Complexity Reduction Schemes for Sphere Decoding

As previously mentioned the use of Schnorr-Euchner (SE) enumeration scheme and the radius reduction technique can result in reduced complexity SD algorithms, without compromising the exact ML or MAP error performance. Efficient algorithms which perform the SE enumeration without the need of *Full Enumeration and Sort (FES)* will be presented in Chapter 6. Several algorithms and techniques have been proposed to reduce the computational complexity of the sphere decoding algorithms, either resulting in a performance degradation or not.

Sorting and Regularization

A common approach to reduce the complexity in the sphere decoding without compromising the performance, is to adapt the detection order of the spatial streams. More efficient pruning of the search tree is obtained if sorting is performed such that "strongest streams", in terms of effective SNR, correspond to levels closer to the root node. For the calculation of a sorted QR decomposition of matrix $\mathbf{H}$, the most common approach is to perform QR decomposition on matrix $\mathbf{HP}$, where $\mathbf{P}$ is a suitably chosen $M_T \times M_T$ permutation matrix. The permutation matrix $\mathbf{P}$ is chosen such that the main diagonal entries of $\mathbf{R}$ in $\mathbf{HP} = \mathbf{QR}$ are sorted in ascending order. The exact solution of this problem has a very high complexity, and several heuristic algorithms have been proposed to reduce the computational requirements of the sorted-QR decomposition pre-processing and the overall sphere decoding complexity [342], [183], [137], [144].

Poorly conditioned channel realizations $\mathbf{H}$ lead to high search complexity due to the low effective SNR on one or more spatial streams. An efficient approach to counter this problem is to operate on a *regularized* channel matrix by computing the sorted QR decomposition of

$$\begin{bmatrix} \mathbf{H} \\ \alpha \boldsymbol{I}_{M_T} \end{bmatrix} \mathbf{P} = \mathbf{QR} \tag{5.28}$$

where α is a suitably chosen regularization parameter. The regularization of the channel matrix results in performance degradation due to the fact that approximations of the max-log LLRs should be used and an additional post-processing step is required after the detection, for the reordering of the LLRs due to the permutation induced by matrix $\mathbf{P}$. Efficient selection of the regularization parameter α for specific channel condition can reduce the performance degradation of the sphere detection with the cost of extra complexity. Several sphere decoding algorithms use this complexity reduction approach with a small performance degradation [339], [343], [318], [287], [286].

Run-time Constraints

As mentioned above, the computational complexity and the decoding throughput of an exact ML or MAP performance, sphere decoding algorithm is variable, which constitutes a problem in many practical application scenarios. Depending on the realization of the random channel matrix as well as on the noise realization, computational requirements of the sphere decoding algorithm can correspond to these of an exhaustive search over the entire set of symbols, in a worst-case scenario. In order to meet the practically important requirement of a fixed average throughput, the algorithm run-time should be constrained. This leads to a constraint of the maximum detection effort or equivalently to an *early termination* of the sphere decoding algorithm. Clearly, such a constraint will affect, in most of the cases, the error performance of the detector.

The most common early termination criterion is the total number of visiting nodes, in the process of the tree traversal. This run-time constraint can be applied in all sphere decoding algorithms for complexity reduction and increased detection throughput [267], [155]. A more efficient approach is to use a variable constraint on the total visiting nodes, based on channel conditions and targeting decoding latency. The estimated channel noise (SNR) can be transformed to a minimum Euclidean distance, in terms of sphere decoding, in which the detection will be terminated (early). The selection of the minimum distance is a trade-off between the detection latency/throughput and the error-rate performance [32], [87].

While SD algorithms are based on symbol-by-symbol detection, the maximum visiting nodes termination criterion can be applied to a block of symbols, based on the decoding block of the channel decoder. This block-based constraint technique schedules a specific maximum value (maximum visiting nodes), for the current symbol detection, based on the previous assigned constraints (for previous symbols), the targeting block-based detection latency and maybe the channel conditions for the current symbol. Furthermore, in the case of SISO MIMO detection this scheduling algorithm can provide variable constraints for the multiple decoding iterations. This approach is more efficient, in terms of error-rate performance and still can guarantee fixed block detection throughput [30], [99].

Another early termination criterion, which is used in complexity reduction schemes for sphere detection algorithms, is the *stopping radius*. In this approach, the current ML or MAP metric is compared with a specific minimum value for triggering the early termination of the decoding process. The stopping radius can be variable in a symbol or block or packet basis, and its value is crucial for the decoder error-rate performance and throughput. An efficient selection, in terms of decoding error performance, is the value of the packing radius [74] of the lattice, for a specific sphere decoder configuration (number of antennas and constellation) [255], [254], [256].

In case of SISO MIMO systems, a run-time constraint can be applied for the early termination of the iterative decoding process. The most simple approach, in terms of complexity, is the *Sign Comparison Approach (SCA)*, in which the signs of the current calculated LLRs are compared with these of the previous decoding iteration [337], [266], [211]. Another technique for early termination of the iterative process is the *Cross Entropy Approach (CEA)*, which has increased computational complexity, compared to SCA, but better error performance [89]. A more efficient technique, in terms of complexity and performance, is the *Performance Driven Approach (PDA)*, in which the termination of the iterative process is triggered when the evaluated bit-error rate (BER) stops significantly to improve, over the iterations [145], [154], [283]. An extension to the PDA technique is proposed in [213], in which the iterative process is stopped when a target BER is achieved.

LLR Clipping

In cases of soft-information SD algorithms, the dynamic range of LLRs is typically not bounded. However, practical systems need to constrain the magnitude of the LLR values (by a maximum value L_{max}) to enable fixed-point implementation. This *LLR clipping* will lead to a performance degradation, however it can be implemented into the SD tree traversal algorithm leading to a reduction in the search complexity. The LLR clipping technique can be used in the sphere decoder for a more aggressive tree pruning in the tree traversal process. The LLR value for a specific bit $x_{j,b}$ is proportional to the difference in (squared) distance between the ML point and the corresponding counter-ML point of that bit. Therefore,

$$\left|LLR\left(x_{j,b}\right)\right| = d_{j,b}^{\overline{ML}} - d_{j,b}^{ML} \leq L_{max} \;\; \Rightarrow \;\; d_{j,b}^{\overline{ML}} \leq d_{j,b}^{ML} + L_{max} \tag{5.29}$$

Equation (5.29) effectively means that clipping the LLR values to L_{max} is equivalent to limiting the search space of the counter-hypothesis points to a sphere of squared radius $d_{j,b}^{ML} + L_{max}$, around the received point $\hat{\mathbf{y}}$ [192].

This technique results in significant reduction of the vising nodes and therefore increased decoding throughput in the cost of reduced error performance. The LLR clipping can be applied in the majority of the soft information SD algorithms and the selection of the maximum LLR value (L_{max}) is a trade-off between detection complexity and error-rate performance [208], [355], [286], [287]. Efficient approaches select the LLR clipping parameter on a symbol/block/packet basis, considering the channel conditions and targeting detection latency [200]. Furthermore, there are also adaptive radius scaling techniques, which dynamically adapt the radius for the counter-hypothesis search space, considering also the current ML metric [192].

An efficient LLR clipping technique for SISO MIMO systems is proposed in [213]. A target bit-error rate (BER) is used as a performance metric to reduce the complexity of the SISO iterative decoding process, with a per-bit selective soft information update method. This technique allows complexity reduction at early iterations of the decoding process, by avoiding the unnecessary processing that would further increase the reliability of those bits that reach the target BER performance. Several LLR clipping methods are proposed in [213] which together with the soft information update technique, adjust on-the-fly, the complexity

of the receiver to the target BER requirements of the system.

Other Schemes

One of the most computational demanding procedures in the SD algorithms is the enumeration process during the tree traversal. As mentioned before, the most efficient way to visit the tree nodes is by ascending order of their PEDs (Schnorr-Euchner (SE) enumeration). There are several algorithms and techniques which reduce the high complexity of the *Full Enumeration and Sort (FES)* for an efficient SE enumeration and which will discuss in the following chapter. Nevertheless, several complexity reduction schemes for the SD algorithms applied to the enumeration process of the tree traversal.

The Euclidean distance calculations of the enumeration process are complex, in terms of efficient implementation, and can be approximated with more simple computations. This will result in more efficient SD implementation, in terms of area and power, with an error-rate performance penalty [29]. Several SE enumeration algorithms take advantage of the geometrical properties of the constellation map to reduce the complexity of the enumeration, without compromising the ML or MAP performance [29], [99]. On the other hand, there are enumeration techniques that approximate the SE with low complexity implementations, resulting in error performance loss, on the average case [196],[326],[167].

Another approach for complexity reduction on the sphere decoding algorithms is the *early pruning* technique. Based on the reduction of the search space of the SD algorithm, these techniques are evaluate and prune the relative less reliable tree nodes before the Euclidean distance calculation process. The early pruning algorithms result in a reduced number of total visiting nodes and therefore increased decoding throughput, with a small (on average) error-rate performance penalty. There are several algorithms for the evaluation of the less reliable tree nodes, which are applied in the initial steps of the tree search or at run-time in the concept of an extra sphere constraint. Finally, the determination of the early pruned nodes is mainly statistical [271], [85], [177].

Chapter 6

Enumeration Schemes on Sphere Decoding

In all the SD algorithms, one of the most demanding processes, in terms of computations, is the node ordering or *node enumeration*. An efficient visiting order of the tree nodes results in reduced decoding complexity due to more aggressive tree pruning. According to Schnorr and Euchner Enumeration (SEE) scheme [259] an efficient way to expand the child nodes is by ascending order of their partial Euclidean distances (PEDs) from the root node. A straight-forward implementation of the SEE requires the calculation and sort of the PEDs of all child nodes, which for the majority of the constellations, have increased complexity due to the large number of child nodes. A parallel structure for all PEDs computations will result to increased chip area, while the sort circuit for large number of values increases the critical path. Therefore, the *Full Enumeration and Sort (FES)* scheme is not efficient for the majority of the MIMO systems.

More efficient schemes should be considered for the SE enumeration, which narrows the SD tree traversal and therefore results in reduced complexity in terms of chip area, power consumption and decoding throughput. An approximation of the Schnorr-Euchner enumeration will have decreased complexity but it will increase the total number of visiting nodes and eventually the total decoding latency, for an exact ML or MAP performance. An efficient enumeration scheme, which consider the geometrical properties of the constellation points, and expand the tree nodes based on the SE enumeration will result in less visiting nodes and reduced decoding complexity without compromising the error-rate performance.

6.1 Introduction

As mentioned before, in MIMO systems with M_T transmit and M_R receive antennas, the symbols s_t ($t = 1, ..., M_T$) from a constellation O of size $|O|$ are transmitted concurrently. In particular, the coded bits are grouped into blocks B_t with the bipolar kth bit $c_k \in \{+1, -1\}$ residing in $B_{\lceil k/\log_2|O|\rceil}$. Then, each block is mapped onto the symbols s_t. The received $M_R \times 1$ vector $\mathbf{y}$ is $\mathbf{y} = \mathbf{Hs} + \mathbf{n}$, where $\mathbf{H}$ is the $M_R \times M_T$ channel matrix that can be QR decomposed as $\mathbf{H} = \mathbf{QR}$ where $\mathbf{Q}$ is a unitary $M_R \times M_T$ matrix, $\mathbf{R}$ an $M_T \times M_T$ upper triangular matrix with elements $R_{i,j}$, and $\mathbf{n}$ is the additive white Gaussian noise vector with samples of variance $2\sigma_n^2$. Maximum a-posteriori (MAP) receivers express the soft information as log-likelihood ratios (LLRs). Using the *max-log* approximation and assuming statistically independent and identically distributed bits, the LLR value of c_k is [318],[281]

$$L(c_k) = \min_{\mathbf{s} \in O_k^{-1}} \{d(\mathbf{s})\} - \min_{\mathbf{s} \in O_k^{+1}} \{d(\mathbf{s})\} \tag{6.1}$$

with

$$d(\mathbf{s}) = \frac{1}{2\sigma_n^2} \sum_{i=1}^{M_T} \left| y_i' - \sum_{j=i}^{M_T} R_{i,j} s_j \right|^2 \tag{6.2}$$

Additionally, $\mathbf{y}' = \mathbf{Q}^H \mathbf{y} = \left[y_1', y_2', ..., y_{M_T}'\right]^T$ and O_k^w are the sub-sets of O having their kth bit equal to w, for $w = +1, -1$. Equation (6.1) shows that the *max-log* LLR calculation can be reformulated into two minimization problems (i.e., hard output SDs) over the different symbol subsets (i.e., O_k^w), per decoded bit. Each minimization problem has its tree root at level $i = M_T + 1$ and leaves at $i = 1$. The $d(\mathbf{s})$ values are calculated recursively: the partial distance (PD) of the $\mathbf{s}_i$ node can be calculated as $D(\mathbf{s}_i) = D(\mathbf{s}_{i+1}) + \Delta(\mathbf{s}_i)$ with $\mathbf{s}_i = [s_i, s_{i+1}, ..., s_{M_T}]^T$ being the partial symbol vectors, $D(\mathbf{s}_{M_T+1}) = 0$ and

$$\Delta(\mathbf{s}_i) = \frac{1}{2\sigma_n^2} \left| y_i' - \sum_{j=i}^{M_T} R_{i,j} s_j \right|^2. \tag{6.3}$$

Applying depth-first tree traversal with SE enumeration and radius reduction [29],[287], with the initial radius assumed infinite, whenever a leaf is reached with its $D(\mathbf{s}_1)$ less than the squared radius r^2, the radius is updated to $D(\mathbf{s}_1)$. Upon meeting a node, if the condition

$D(\mathbf{s}_i) \geq r^2$ holds, then this node, the node's children and the node's siblings not visited yet will be pruned. To define the search order (i.e., the child of node $\mathbf{s}_{i+1}$, to be visited next) SE enumeration [259] is typically employed, according to which the nodes are visited in ascending order of their PDs, or equivalently of their $\Delta(\mathbf{s}_i)$ values. An important advantage of the SE enumeration is that leaves that are more likely to lead to the ML solution are found early, which expedites the pruning of the tree, resulting in reduced decoding complexity. However, the overall SD complexity may not improve due to the increased ordering complexity. An example is the soft-output SD with high-order, non-constant amplitude constellations (e.g. 16-QAM), where the SE enumeration is inefficiently realized by fully enumerate and sort (FES) all children PDs, even when few nodes are finally visited due to tree pruning.

To perform SE enumeration for high order constellations without calculating and sorting the PDs of all children nodes (i.e., without FES), state-of-the-art approaches [99],[29] split the given QAM constellation into sub-constellations in such a way that for each sub-constellation ordering can be achieved by using simple geometrical properties (i.e., in a "zig-zag" order), which also leads to the straightforward and low-complexity choice of the best candidates in each subset. In particular, the QAM constellation is divided into several phase-shift-keying constellations (PSK-wise splitting) or several pulse-amplitude-modulation constellations (PAM-wise splitting) and the search order is calculated by exhaustive search over these constellation subsets. Hence, the enumeration computational complexity and the critical path of the corresponding implementations, is improved with respect to FES. While such approaches result in a reduced number of redundant calculations and a reduced sorting order, still unnecessary PD calculations are performed.

Approximate SE ordering methods [196], [326], [167], [330], [351] can provide performance very close to the optimal for specific scenarios with significant gains in terms of complexity. These methods use mainly a partial predefined ordering or a simplified metric, in comparison with the Euclidean distance, for the ordering of the child nodes, resulting in low complexity implementations with reduced critical path. On the other hand, the approximated SE enumeration can lead to inefficient tree pruning and increased number of total visiting nodes during MIMO detection, while resulting in error-rate performance losses. An appropriate modification of the pruning metric should be considered, for such solutions, to

be able to guarantee the exact *max-log MAP* performance for any operational environment and performance metric (e.g., bit-error-rate, packet-error-rate, etc).

The hardware implementation of a SD algorithm should maximize the resource utilization and avoid redundant computations. In the case of a depth-first tree traversal strategy, this can be achieved when the decoder visits a new node in each cycle and when no node is ever visited twice. In [29] an efficient hardware architecture is proposed for the depth-first SD algorithm, the *One-Node-Per-Cycle (ONPC)* architecture. The main two modules of this architecture is the *Metric Computation Unit (MCU)*, which is responsible for the forward recursion of the tree traversal and the *Metric Enumeration Unit (MEU)*, responsible for the backward recursion. The enumeration process of the SD algorithm is implemented in the MEU module and the ONPC architecture requires a new enumerated node at each clock cycle. An asynchronous implementation of the enumeration process inside the MEU, should be consider for an efficient hardware implementation of the SD algorithm. Therefore, the complexity of the enumeration scheme used in a SD implementation is critical for the decoder throughput, latency and power consumption.

6.2 PAM-wise Enumeration Scheme

As mentioned above, the PAM-wise enumeration scheme divides the constellation points in subsets based on the row or column they belong [99]. The closest constellation point to a received point r_i can be identified with low complexity computations (boundaries checks - *slicer*). Fig. 6.1 shows a 16-QAM constellation map and the resulting PAM subsets. The closest point to the received point r_i is the constellation point A_0. With a column-wise partitioning of the constellation map the point A_0 is assigned to the first subset (subset A) along with the points A_1, A_2 and A_3. The first best candidate of each subset is the constellation point that belongs to the same row as the point A_0, as illustrated in the Fig. 6.1 with red circles (B_0, C_0 and D_0).

To identify the first preferred child (among A_0, B_0, C_0 and D_0), a computation of the four PD metrics $\Delta(s_i)$ should be performed. The constellation point with the smaller PD metric (A_0 in Fig. 6.1) is then selected as the preferred child node, only if its PD metric $D(s_i) = D(s_{i+1}) + \Delta(s_i)$ is smaller than the sphere constraint r^2. The next best candidate

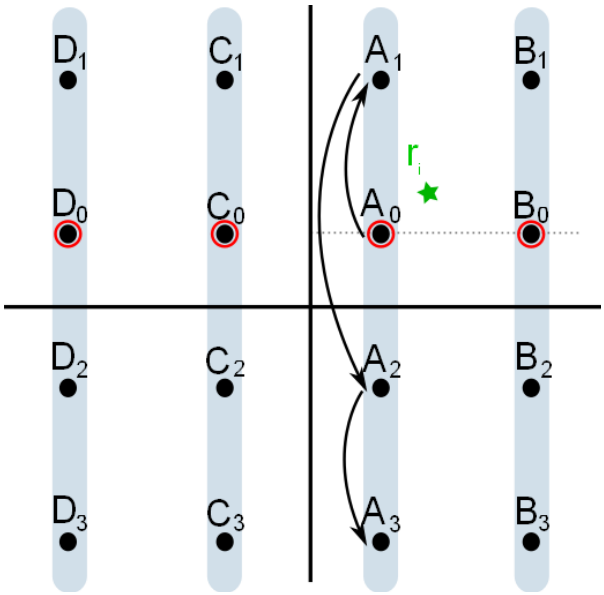


Figure 6.1 PAM-wise enumeration scheme.

of the subset A is determined by applying the zig-zag order as shown in the figure (point A_1). The direction of the zig-zag ordering (for all subsets) is determined by the position of the received point r_i in comparison to the line which passes from constellation points which are in the same row as the closest constellation point (A_0, B_0, C_0 and D_0), as illustrated in the Fig. 6.1. This can be also calculated by low complexity boundaries checks. The next preferred constellation point is determined by the smaller PD metric ($D(s_i)$) among the points A_1, B_0, C_0 and D_0 of the constellation, only if this metric not exceeds the sphere constraint (r^2). The next best candidate of the subset with the preferred constellation point, is determined with the zig-zag ordering, with the same direction.

The PAM-wise enumeration technique requires reduced number of PD calculations compare to FES. At the initialization step of the method four PD calculations are performed, one for each candidate, assuming a 16-QAM constellation map. For each of the steps following the initialization, the technique performs three operations: i) a PD computation, if this is needed, to identify the next best candidate, ii) select the minimum among four PD metrics one from each subset and iii) test it against the sphere constraint to determine if the selected constellation point has to be pruned. Thus, the method requires at least four PD computations and a selection circuit for the identification of the smaller PD metric, regardless the sphere constraint r^2.

Implementation

As mentioned above, in the case of 16-QAM constellations, the specific enumeration technique divides the constellation map in four subsets. Fig. 6.2 shows the top level architecture of the PAM-wise enumeration implementation. The received symbol r_i allows us to determine the closest constellation point (*CP*) and the zig-zag order direction (*dir*), by applying low complexity computations. The component computing the candidates for each subset requires one memory bank (ROM) for storing the indices of the constellation point's row. We have implemented the PAM-wise technique by letting the four points of each subset to belong to a distinct column, as shown in Fig. 6.1. Therefore, we can reduce the total number of ROMs, to only one memory for all subsets because according to the above mapping, each of the subsets uses the same indices to identify the constellation point's row and the same zig-zag order direction. Hence, for any of the four subsets, the indices of the constellation point's row will be read in the same sequence determined by the zig-zag order direction.

The top-level architecture also includes four 16-QAM ROMs, one for each subset. Each of the QAM ROMs includes the IQ values of the constellation points sorted based on their constellation index, which is used as the ROM address. Furthermore, a selection circuit decides the next best candidate, for each of the subsets. This module is actually a set of four (4) independent multiplexer circuits. These circuits are controlled by the output of a 4-bit decoder, the input of which is the 2-bit *selection* signal of the *PD-compare* unit, which indicates the subset with the selected constellation point, among the four candidates. The output of the 4-bit decoder module will trigger the selection of the new candidate point, based on the zig-zag direction, only for this subset, while the candidate points of the other subsets will remain the same.

The four *PD-calculation* units compute the PD metrics for each of the candidate points. The *PD-compare* unit selects the point with the smallest PD metric and forwards this metric $\Delta(s_i)$ to the *sphere constraint check* unit. The selected point becomes the preferred child only if its PD metric $D(s_i)$ is smaller than the sphere constraint r^2. The *sphere constraint check* component requires an adder for the calculation of the PD metric $D(s_i)$ and a comparator to decide whether the specific point must be pruned or expanded. If the PD metric $D(s_i)$ is larger than the sphere constraint r^2 then the enumeration process stops, because all the next selected points will have PD metrics larger than r^2.

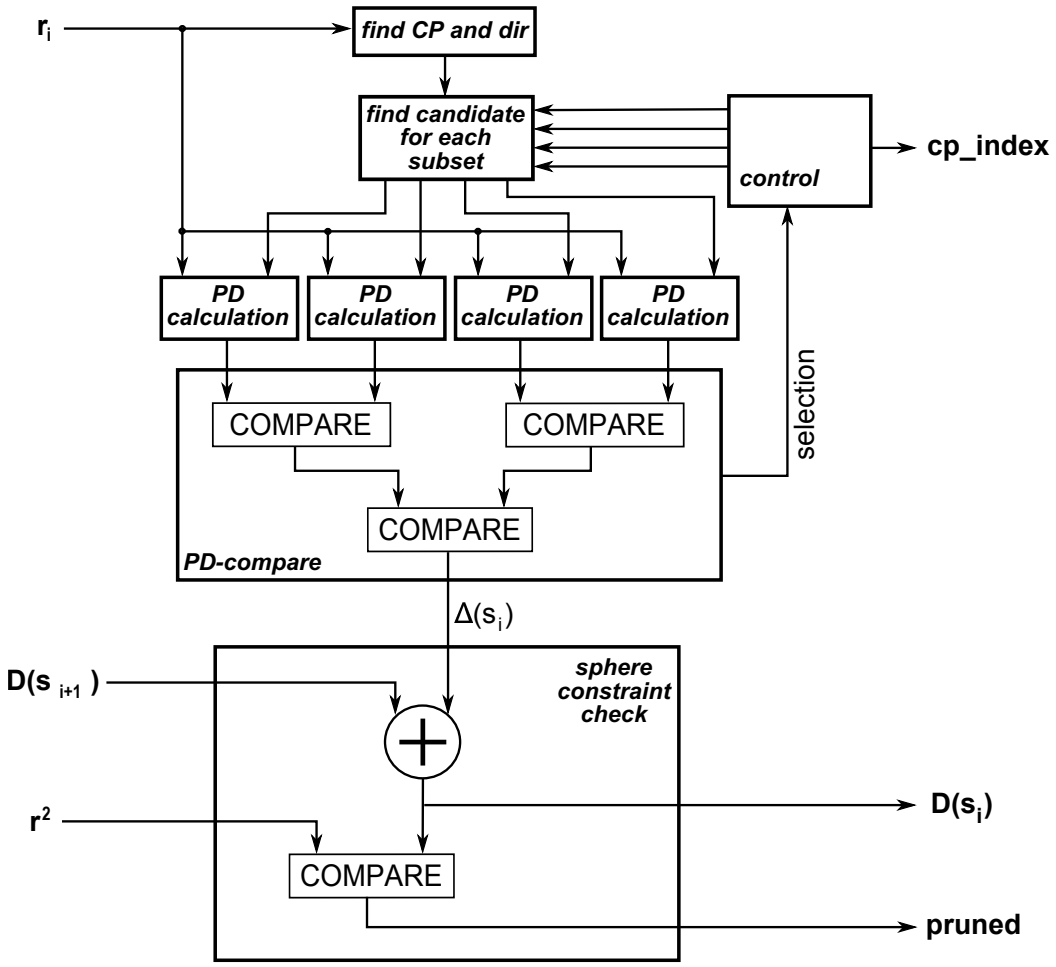


Figure 6.2 PAM-wise Enumeration Technique Architecture.

In the initialization of the PAM-wise enumeration process four PD metrics should be computed, while in normal operation only the PD metric of the next candidate is evaluated. The *PD-compare* unit in every step of the process, selects the smaller of four PD values, while the *sphere constraint check* module is evaluate and compare the metric $D(s_i)$ with the value of r^2. Due to the complex multiplication units inside the *PD calculation* modules, the significant portion of the power is consumed for the computation of the PD metrics. For small sphere constraints, only one or two child nodes can be visited, while the initialization process of the PAM-wise enumeration scheme, evaluates four PD metrics. This results in high power consumption for specific cases, compared to an enumeration scheme with only one PD calculation in the initialization process.

6.3 PSK-wise Enumeration Scheme

In the case of PSK constellations, the points form a complex circle in the IQ space. An efficient enumeration scheme, for these constellations, is the zig-zag technique after the identification of the closest constellation point, to a received symbol r_i, and the zig-zag direction. In the QAM constellations we can split the points in multiple concentric circles (subsets) and perform PSK-wise enumeration, after the identification of best candidate point and zig-zag direction for each of the subsets [105], [169], [29].

In [105], complex trigonometric functions are used, and the identification of the closest point, to a received symbol r_i, and the zig-zag direction (dir) is calculated based on the angle of the received symbol r_i and the angles of the constellation points. The proposed method in [169] is similar, with the used of predefined radius tables and sliced versions of the PSK-wise subsets, based on the received symbol, to reduce the computation complexity. Authors in [29] propose an efficient technique for the identification of the closest point and the calculation of the zig-zag direction for the PSK-wise enumeration scheme in PSK or QAM constellations. This technique is based on multiple simple boundary checks to avoid the costly trigonometric calculations of [105].

Figure 6.3 shows the three subsets of a 16-QAM constellation map. Each subset have different number of points, in comparison with the PAM enumeration, in which all subsets have the same number of constellation points. Following the same notation, as in the case of PAM enumeration technique, the constellation points in the Fig. 6.3 are labeled based on the subset they belong (A, B or C). The closest constellation point to a received point r_i is identified with low complexity boundary checks, same as in the case of PAM enumeration scheme. In Fig. 6.3, A_0 is the closest point to r_i. The candidates for the other two subsets are selected based on the position of the r_i in comparison with the line $x = y$, where x is the real component of the points and y is the imaginary, as illustrated in the figure [29]. The points B_0 and C_0 are the candidate constellation points from subsets B and C, as shown in the figure with red circles.

On the other hand, the zig-zag direction is not common among the constellation subsets as in the case of PAM-wise enumeration scheme. For each subset a different calculation is performed for the identification of the zig-zag direction. As shown in Fig. 6.3, the computation of the zig-zag direction of the subsets A and C is based on the comparison between

the real and imaginary components of the r_i, while the calculation of the direction for the subset B has additional shift operations. Furthermore, Fig. 6.3 shows the zig-zag directions for the three subsets.

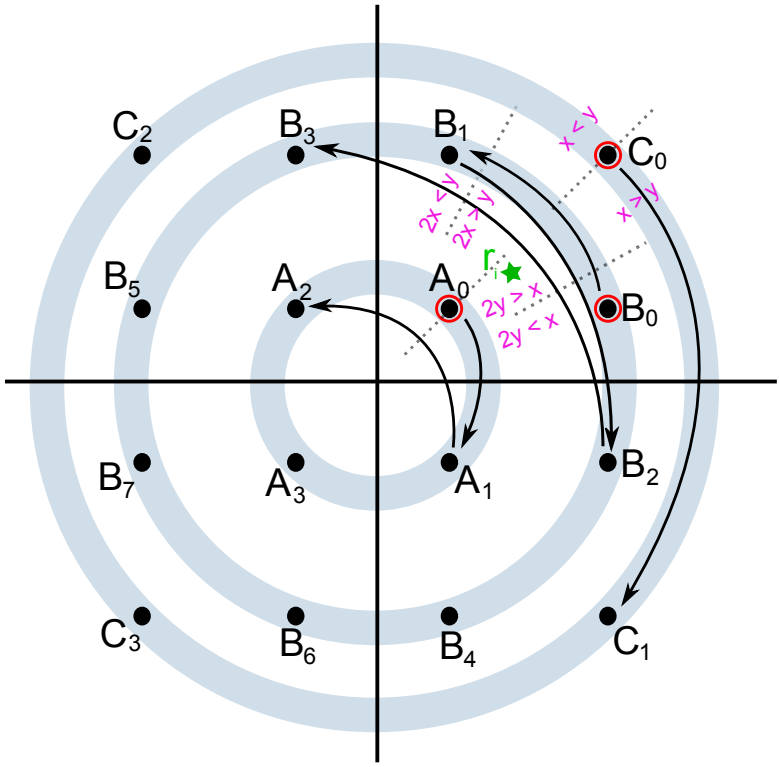


Figure 6.3 PSK-wise enumeration scheme.

As in the case of PAM-wise enumeration technique, the PD metrics of the three subset candidates (A_0, B_0 and C_0) are calculated and the constellation point with the minimum PD metric $\Delta(s_i)$ is the preferred child, only if the sphere constraint is not compromised ($D(s_i) = D(s_{i+1}) + \Delta(s_i) < r^2$). Based on the zig-zag direction of the specific subset the next candidate constellation point (A_1) is selected and its PD metric is calculated and compared with the other two (B_0 and C_0).

In comparison to the PAM-wise enumeration scheme, the PSK-wise technique requires one subset less, for the case of 16-QAM constellation, while in the case of 64-QAM constellations, PSK-wise has one subset more than the PAM-wise scheme. Furthermore, more complex calculations are performed in the PSK-wise enumeration technique for the identification of the closest constellation point and zig-zag direction and for the case of larger QAM constellations (e.g. 64-QAM) the increased computational complexity is significant.

The possibility of different zig-zag direction per subset results in complex control structures, which for larger constellation may result to an increased overhead in the critical path, of the enumeration process.

Implementation

The implementation of the PSK-wise enumeration scheme for the case of 16-QAM constellations is based on three subsets. Figure 6.4 shows the top-level architecture of the PSK-wise enumeration module.

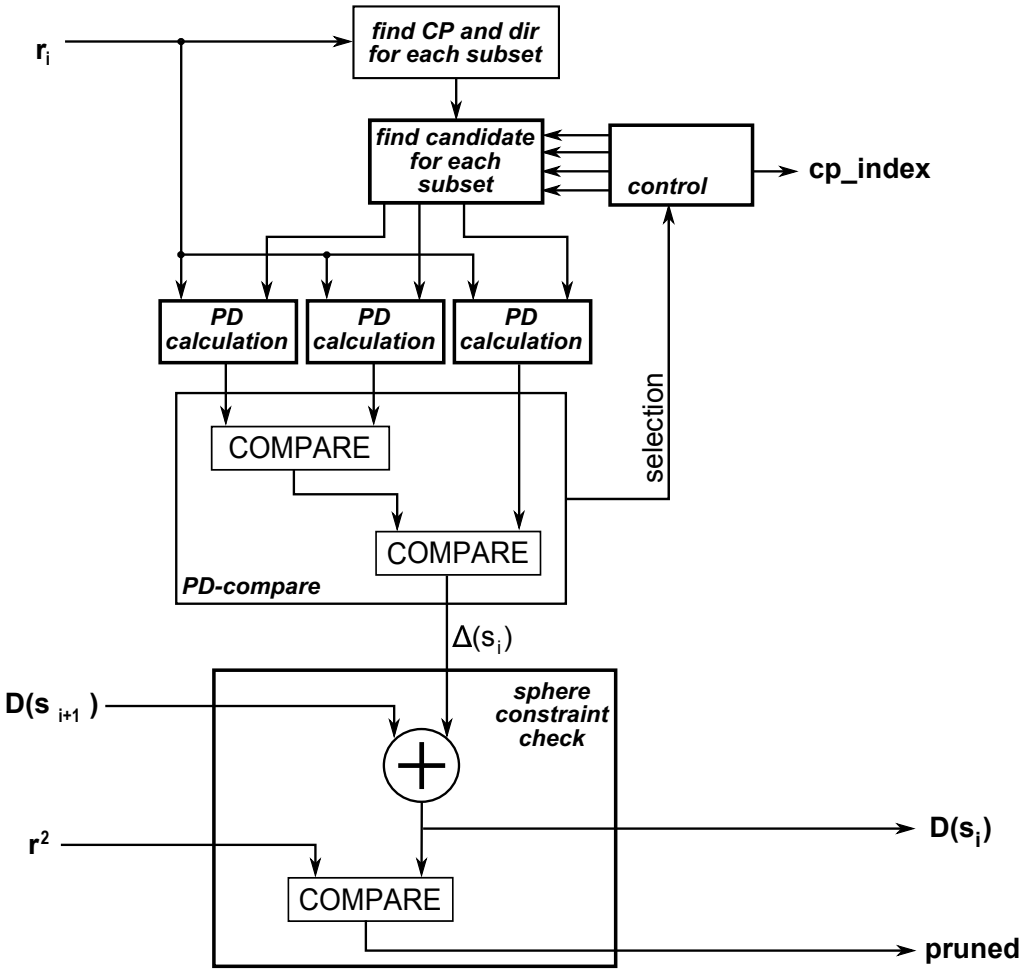


Figure 6.4 PSK-wise enumeration scheme.

The received symbol r_i is forwarded in the first module for the identification of the closest constellation point (*CP*) and zig-zag direction (*dir*), for each of the three subsets. The low complexity boundary checks involve also shift operations, for the case of 16-QAM re-

sulting in increased complexity compared to the PAM-wise enumeration architecture. Each of the subsets include different number of total constellation points and zig-zag direction, which result in limited optimizations regarding the subset *Look-up Tables (ROMs)*. The calculated zig-zag direction will affect the reading sequence of the ROMs, during the enumeration process.

As for the PAM-wise enumeration technique, the top-level architecture of the PSK-wise enumeration scheme includes three 16-QAM ROMs, one for each of the subsets. Furthermore, a similar selection circuit is used for the evaluation of the next best candidate, for each of the subsets. The PD metrics of the three candidate points is evaluated with the *PD calculation* units and the *PD compare* unit is responsible to find the smaller PD metric. It includes one comparator unit less than the corresponding unit of the PAM-wise architecture, due to reduced number of subsets, while the critical path of the module is the same. The metric $D(s_i)$ is calculated and compared with the sphere constraint (r^2), for the selected candidate point, to determine if this child node should be expanded or pruned. If the metric $D(s_i)$ for a specific node is larger than the sphere constraint the enumeration process is stopped, since all the next selected points will have PD metrics larger than r^2.

The initialization process of the PSK-wise enumeration scheme includes three PD computations, in comparison to four of the PAM-wise enumeration. The critical path of the PSK-wise architecture is similar to that of the PAM-wise, with the exception of the boundary checks module for the evaluation of the zig-zag directions, for the subsets. Nevertheless, the increased power consumption for specific cases, with small sphere constraints, is similar to the PAM-wise architecture due to unnecessary PD computations in the initialization process of the technique.

6.4 Other Enumeration Schemes

Several enumeration schemes have been proposed in the literature, to overcome the high complexity of the PAM-wise and PSK-wise enumeration techniques. Most of these approaches use predefined visiting order, based on geometrical characteristics of the constellation map, to approximate the SE enumeration. This approximations result in increased number of visiting nodes and losses in the decoding performance, while the reduced enu-

meration complexity results in power savings for the tree traversal process.

Search Sequence Determination

An efficient approximate SE enumeration is proposed in [196], which is based on bisector lines on the constellation lattice. The *Search Sequence Determination (SSD)* technique uses simple boundary checks for the determination of the closest constellation point (CP), to a received symbol r_i. These boundary checks form a square area around the CP and further bisector lines, in this square area, can be used for the sequence determination of the constellation points.

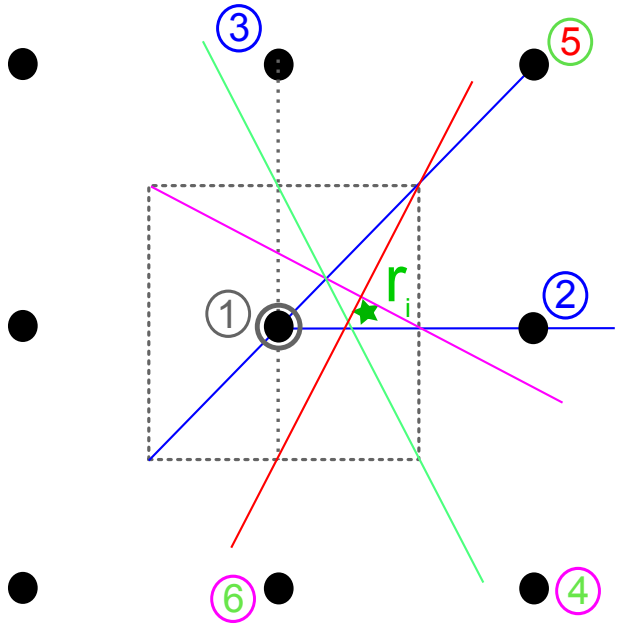


Figure 6.5 Search Sequence Determination (SSD) enumeration scheme.

Based on the quadrant around the CP, in which the received symbol is lying, a specific bisector line can be used to enumerate the first two nodes based on the SE enumeration. This line form two triangular areas in the specific quadrant and by identifying in which triangular the r_i is lying, the enumeration of the first two nodes can be determined, as shown in Fig. 6.5 (blue line). Simple comparisons of the real and imaginary parts of the r_i can be used for the enumeration of the first two constellation points. Based on bisector lines spanned by the nodes, the enumeration of the first seven constellation points can be performed (Fig. 6.5) with simple calculations, including additions, shifts and comparisons. The sequence

determination for the rest of the constellation points results in more complex computations, including multiplications and several comparisons, due to the increased number of possible nodes.

An approximation technique is used in [196] to avoid the high complexity of the SSD enumeration for high-order nodes (above seventh node). Predefined sequences stored in LUTs are used, based on the reference location of the received symbol r_i, compared to the closest constellation point (CP). The memory requirements and the enumeration accuracy are relative to the total number and the reference location of the predefined sequences. The implementation in [196] uses the triangle's centers as sample reference point for the proposed predefined SSD enumeration technique. The LUT-based approximation of the sequence determination results in enumeration errors (compared to SE enumeration) for all nodes after the third preferred constellation point. These errors could increase the total number of visiting nodes and reduce the decoding throughput, while resulting in performance degradations.

The implementation of the SDD enumeration scheme includes simple calculations for the first enumerated nodes, while high-complexity computations are required for the enumeration of the rest of the constellation points. The proposed predefined sequence determination approach, avoids the complex calculations while introducing additional errors in the enumeration process. Trade-offs between the number and size of the LUTs and the enumeration accuracy should be considered for the proposed approximate SE enumeration technique.

LUT-based Enumeration

An approximate SE enumeration technique with low-complexity implementation is presented in [330] and [351]. The constellation map is divided in sub-regions and for each sub-region a pre-calculated visiting order is stored in a *Look-Up Table (LUT)*. Each memory address of the LUT includes the enumerated constellation points in ascending order of their partial distances, from the centers of each sub-region. By exploiting the constellation symmetries a reduction of the total memory requirements is feasible. An increased number of constellation sub-regions results in more accurate node enumeration and reduced number of visiting nodes, with the cost of increased complexity due to multiple ROMs and more

complex control.

A more efficient LUT-based enumeration scheme is the *Tabular Enumeration*, presented in [167]. After the identification of the closest constellation point (*CP*), the region around the CP is divided into eight triangular sub-regions. For each sub-region, the most likely visiting order of all other constellation points is computed in advance and stored in LUTs. Direct implementation of the tabular enumeration technique requires eight tables for each constellation point, each with $2^Q - 1$ entries, assuming a 2^Q-QAM constellation. Based on the symmetry of the eight sub-regions and the shift invariance property of the partial distance calculation, a reduction of the memory requirements is feasible. A simplified version, of the generic tabular enumeration, is presented in [167], for efficient node ordering with best-first tree traversal SD algorithms. The tabular enumeration scheme is a low-complexity node ordering implementation with the disadvantage of increased total number of visiting nodes and performance degradation, due to the approximate SE enumeration.

Approximate l^∞-norm SE enumeration

An efficient implementation of an approximate SE enumeration scheme is proposed in [326]. The specific technique is based on simplified approximate distance calculations, without the need for predefined visiting order stored in LUTs and therefore, scales well to higher-order constellations. The low-complexity l^∞-norm is used for the nodes enumeration, where $l^\infty(x) = \|x\|_\infty = max(|Re(x)|, |Im(x)|)$, $Re(x)$ and $Im(x)$ denote the real and imaginary part of the entries of x, respectively.

The identification of the closest constellation point (CP) is based on a slicer, while the region around the CP is divided in eight sub-regions. Simple geometric rules are used to define the sub-region containing the received symbol r_i, while the constellation points are enumerated according to their l^∞-norm. The points with the same l^∞-norm form an one-dimensional subset, as shown in the Fig. 6.6, while all nodes within the same subset are processed before the algorithm selects the next subset. The selection of the constellation points contained in the same subset is based on the zig-zag ordering, around the closest constellation point.

While the l^∞-norm is used only for the enumeration of the nodes and not for the distance calculations, the proposed method is an approximate SE enumeration resulting in increased

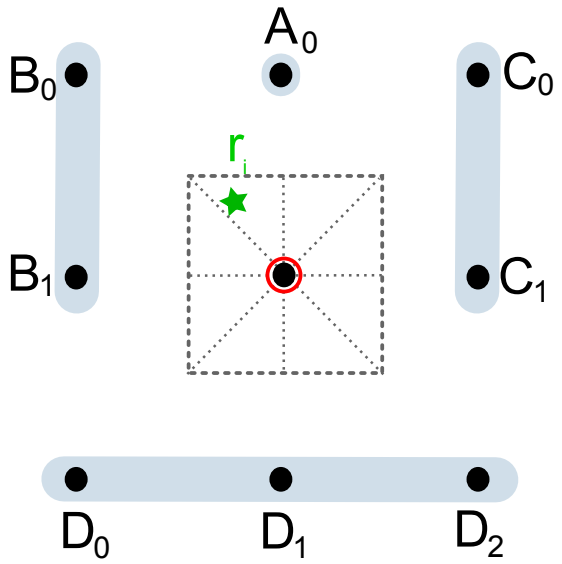


Figure 6.6 Approximate l^{∞}-norm enumeration scheme: Constellation points subsets based on their l^{∞}-norm distance.

number of visiting nodes and error performance degradation. The implementation complexity of this enumeration process is low, compared to other approximate SE enumeration schemes, especially for large constellations (constellations including and beyond 64-QAM).

2D Zig-Zag Complex Enumeration

An exact SE enumeration scheme is proposed in [264],[262], [191], for the case of K-best breadth-firth SD algorithm, with complex QAM constellations. Based on the observation that in a PAM constellation the zig-zag visiting order of the nodes is an exact SE enumeration, the proposed method uses this visiting order on both vertical and horizontal directions, in parallel.

After the identification of the closest constellation point, with a simple slider, the proposed technique visits the nodes by performing zig-zag ordering on both directions. The partial Euclidean distances (PEDs) for the visiting nodes are calculated and stored in a priority queue. The node with the smaller PED in the priority queue is the next visiting node for the tree traversal. As shown in the Fig. 6.7 the first enumeration step, after the CP identification (node a), visits the nodes b and c, by performing horizontal and vertical zig-zag enumeration. The PEDs for the two nodes are evaluated and stored in the priority queue L. The second enumeration step (Fig.6.7b) selects the node b as the next visiting node for the

tree traversal and performs only vertical zig-zag enumeration (node d), from the selected node b. The horizontal zig-zag enumeration from an already visited node which belong to the same column as a previously selected node, is bypassed by the technique to avoid visiting a node (evaluate and store the PED into L) more than once [264], [262].

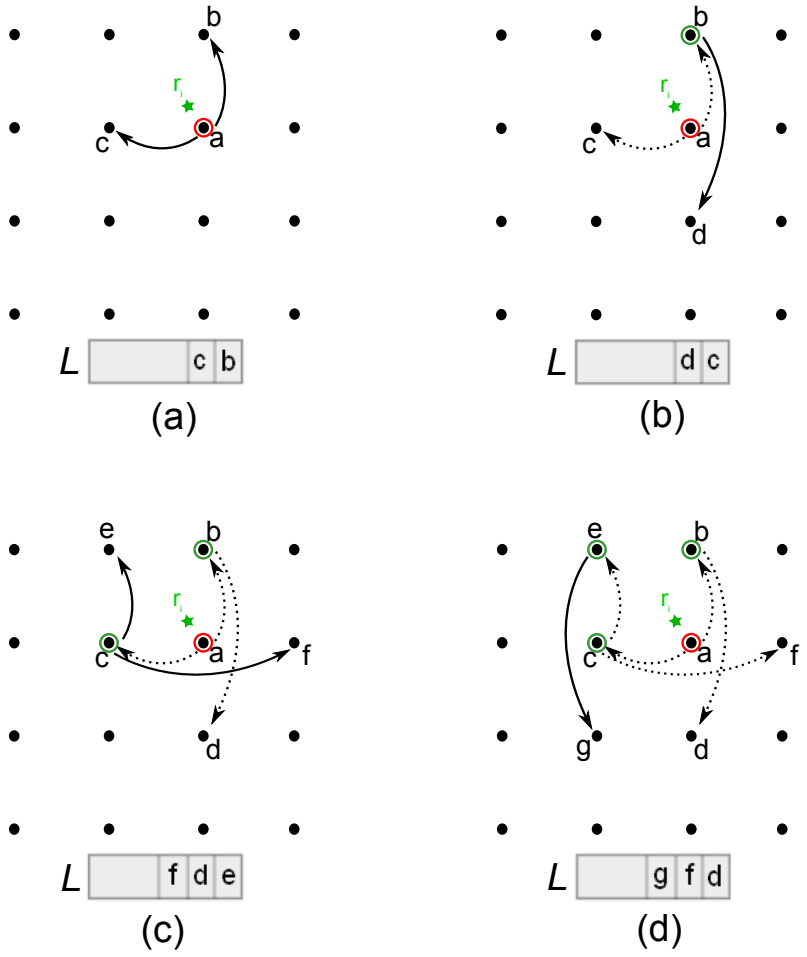


Figure 6.7 2D Zig-Zag Complex Enumeration scheme: enumeration steps for the first three constellation points. Priority queue is shown at the bottom of each enumeration step.

The third enumeration step at Fig.6.7c, selects the node with the smaller PED in the priority queue L (node c) and performs horizontal and vertical zig-zag enumeration (nodes e and f). At the last enumeration step, shown at Fig.6.7d, the algorithm selects the node e as the node with the smaller PED in the priority queue L, and performs only vertical zig-zag enumeration (node g) as a previously selected node (node c) belongs to the same column as the current selected node.

As shown in [262] the 2D zig-zag complex enumeration technique performs SE enu-

meration by selecting the nodes in ascending order of their PEDs from a received symbol r_i. Furthermore, the constraint of bypassing the horizontal zig-zag enumeration from a selected node, when a previously selected node belongs to the same column, ensures that each node will be visited only once, by the enumeration method. The maximum length of the priority queue is $\sqrt{|Q|}$ for a 2^Q-QAM constellation map [262]. A similar enumeration method is used in [217] for the case of depth-first SD algorithm, with the addition of a geometrical tree pruning technique to increase the decoding efficiency.

The specific enumeration process can be efficiently scaled to dense and very dense constellations [81] due to the fact that requires a maximum of two PD computations in parallel, independently of the constellation size. On the other hand, the required constraint for bypassing the horizontal zig-zag enumeration results in high control complexity, while the priority queue of size $\sqrt{|Q|}$ increase the critical path of the 2D zig-zag complex enumeration implementation architecture, resulting in decreased decoding throughput.

6.5 Advanced Enumeration Scheme

This section presents a novel low-complexity enumeration scheme, which is based on a predefined approximate visiting order, which can guarranty the *ML* or *max-log MAP* performance of the MIMO decoder [214]. Instead of the exact SE enumeration a predefined visiting order is used by the method, to reduce the computational complexity. While the SE enumeration visits the symbols in ascending order of their $\Delta(\mathbf{s}_i)$, the proposed method uses a lower limit of the $\Delta(\mathbf{s}_i)$, which can be pre-calculated by exploiting geometrical characteristics of the QAM constellation. Based on the aforementioned node ordering and by modifying the pruning metric, the proposed enumeration scheme can guaranty the *ML* or *max-log MAP* performance with the cost of increased number of visiting nodes. However, we can relax the requirement for guaranteed optimality with negligible performance loss, as will discuss later in detail.

Node Visiting Order

Instead of sorting $\Delta(\mathbf{s}_i)$ from (6.3), we can use:

$$\Delta(\mathbf{s}_i)\frac{2\sigma_n^2}{|R_{i,i}|^2} = |r_i - s_i|^2 = \delta^2(\mathbf{s}_i) \tag{6.4}$$

with

$$r_i = \frac{y_i' - \sum_{j=i+1}^{M_T} R_{i,j}s_j}{R_{i,i}} \tag{6.5}$$

Based on (6.4) the SE enumeration process visits the tree nodes by ascending order of their $\delta^2(\mathbf{s}_i)$. The proposed enumeration scheme uses a tight lower limit of the $\delta^2(\mathbf{s}_i)$ which is the $\delta^2_{min}(\mathbf{s}_i)$, which can be pre-calculated based on geometrical characteristics of the constellation map.

Low-complexity calculations can be used to identify the closest constellation point $s^{(0)}$ to a received symbol r_i, by a typical QAM detector (i.e., "slicer"). The rectangular area with centre the symbol $s^{(0)}$ and sides equal to L, which is the distance of the constellation points, will contain the received symbol r_i, as shown in Figure 6.8 with the shadowed rectangle (ABCD). The circle with centre the symbol $s^{(0)}$ and radius $L/\sqrt{2}$ will also contain the received symbol, as it contains the aforementioned rectangle. All the constellation points can be grouped in concentric circles of radii a_k, with k denoting the index of the group circle. Thus, the distance between the received symbol r_i and any point in the group with radius k, cannot be less than $\delta_k = a_k - L/\sqrt{2}$.

A tighter bound can be calculated based on additional geometrical characteristics of the constellation map. Simple sign comparisons of the real and imaginary part of the $s^{(0)}$ and r_i can be used to identify the quadrant Q, in which the received symbol r_i lies (1st quadrant in the Fig. 6.8). All the constellation points of the k-th group or circle, which lie in the opposite quadrant $\bar{Q}$ (3rd quadrant in the figure), will have distances from r_i which cannot be smaller than a_k. Therefore, for each symbol s_i of the k-th group holds:

$$\delta(s_i) = |r_i - s_i| \geq \delta_{\min}(s_i) = \begin{cases} a_k - L\Big/\sqrt{2} & ; s_i \notin \bar{Q} \\ a_k & ; s_i \in \bar{Q} \end{cases} \tag{6.6}$$

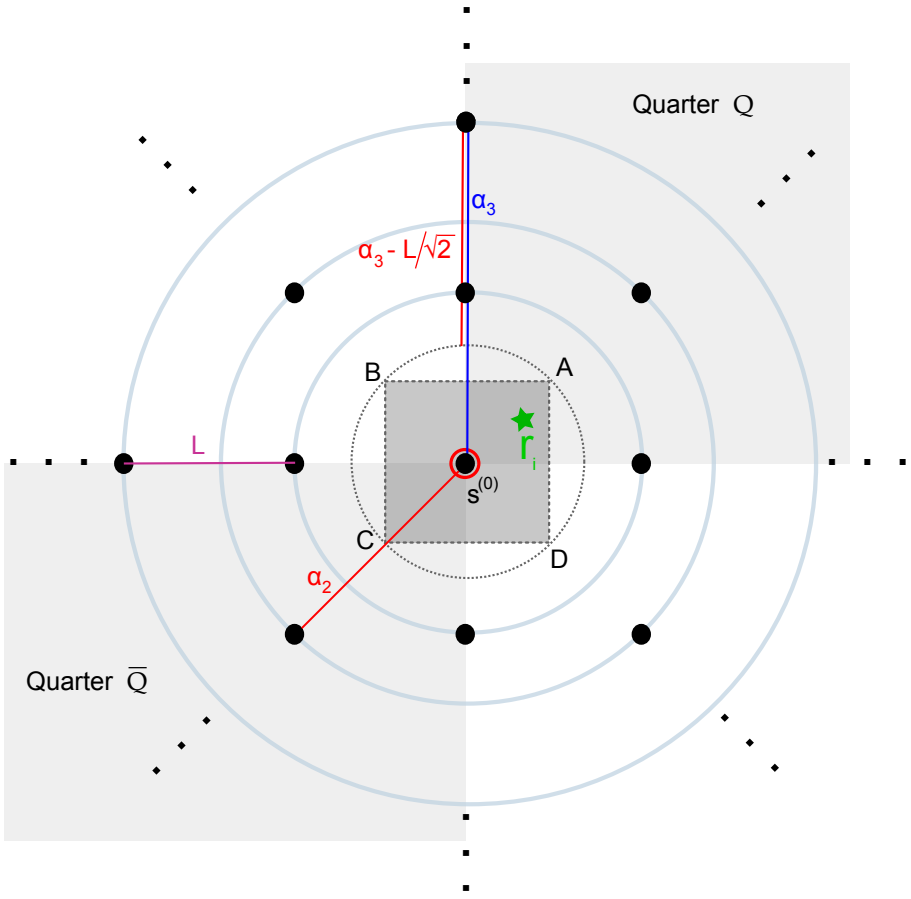


Figure 6.8 Example showing the geometrical characteristics of the constellation which are exploited by the proposed enumeration approach

The boundary case of a received symbol lying outside the constellation map, it is also covered by (6.6), since the distance to all symbols of the k-th circle cannot be less than the distance provided by (6.6). We can enumerate the nodes based on the ascending order of their $\delta_{\min}(s_i)$, or equivalently of their $\Delta_{\min}(\mathbf{s}_i)$.

Tree Pruning

Enumeration schemes which use a predefined visiting order have been proposed in the literature [196], [330], [351], [167]. These approaches can not guarantee *ML* or *max-log MAP* performance due to the approximations of the predefined ordering. The proposed enumeration scheme avoids compromising the *ML* or *max-log MAP* performance by modifying the

pruning metric accordingly. Since $\delta^2_{\min}(\mathbf{s}_i) \leq \delta^2(\mathbf{s}_i)$ we have:

$$D_{min}(\mathbf{s}_i) = D(\mathbf{s}_{i+1}) + \delta^2_{\min}(\mathbf{s}_i)\frac{|R_{i,i}|^2}{2\sigma_n^2} \leq D(\mathbf{s}_i) \tag{6.7}$$

and if $D_{min}(\mathbf{s}_i) \geq r^2$, or equivalently if

$$\delta^2_{\min}(\mathbf{s}_i) \geq \frac{r^2 - D(\mathbf{s}_{i+1})}{|R_{i,i}|^2} 2\sigma_n^2 = \tilde{r}^2 \tag{6.8}$$

it will also hold that $D(\mathbf{s}_i) \geq r^2$. As mentioned above we will visit the nodes in ascending order of their $\delta^2_{\min}(\mathbf{s}_i)$ and thus we can safely prune this node, all its non-visited siblings (*horizontal pruning*) and children nodes (*vertical pruning*), without compromising the *ML* or *max-log MAP* performance of the sphere decoder, with the extra cost of additional visiting nodes, compared to the SE enumeration scheme. Furthermore, if $D(\mathbf{s}_i) \geq r^2$ (and $\delta^2_{\min}(\mathbf{s}_i) < \tilde{r}^2$) then the propose approach will prune all the children nodes but not the siblings (only vertical pruning is performed). Finally if both $\delta^2_{\min}(\mathbf{s}_i) < \tilde{r}^2$ and $D(\mathbf{s}_i) < r^2$, no pruning takes place.

The *ML* or *max-log MAP* performance of the decoding process is guaranteed due to the modification of the pruning metric, with the additional cost of increased visiting nodes, in comparison to the SE enumeration process. An "approximate" version of the proposed scheme is also considered, in which the horizontal and vertical pruning is performed based only on the partial Euclidean distances of the predefined visiting order. This "approximate" version avoids the increased number of visiting nodes, during the detection process, with a negligible performance loss, as we will see in the next subsections.

Implementation

The block level architecture of the proposed enumeration scheme, for both the "exact" and "approximate" versions, is depicted in Fig. 6.9. First, the *find* $s^{(0)}$ *and* Q unit calculates the closest constellation point $s^{(0)}$ and the quadrant Q containing the received symbol r_i, with low-complexity boundary checks. The predefined visiting order of the nodes is stored in the *visiting-order memory* ROM in the *vo_calc* module, which is responsible to provide the next visiting node based on $s^{(0)}$, Q and the value of the *enumeration-counter*. The *control* unit counts the number of enumerated points, for a specific tree level, for the calculation of

the next visiting node and the metric $\delta^2_{\min}(\mathbf{s}_i)$.

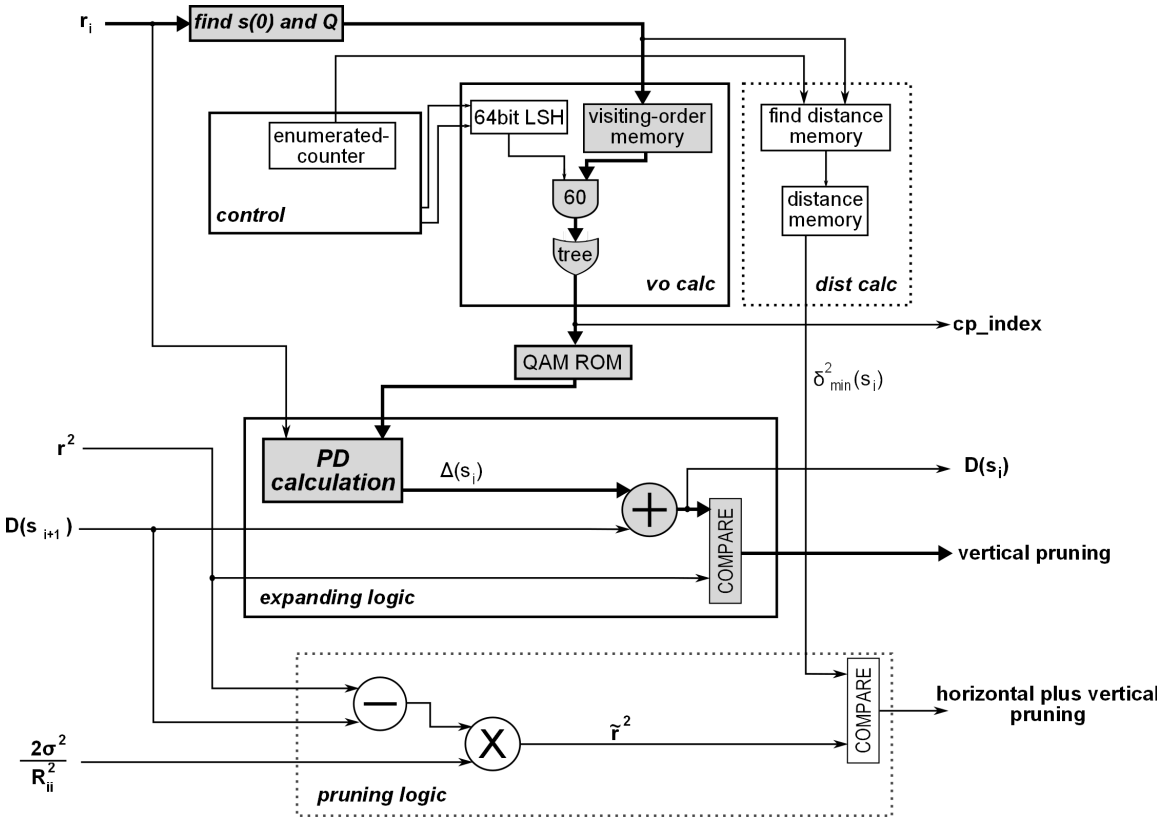


Figure 6.9 Proposed Enumeration Technique Architecture.

The *vo_calc* unit computes the enumerated constellation point indices. It uses the *visiting-order memory* to compute the visiting order of the symbols. Each line of the *visiting-order memory* contains the constellation points indices in visiting order. The address of *visiting-order memory* is the concatenation of the index of the closest point ($s^{(0)}$) and the quadrant Q, of the received symbol r_i. The *vo_calc* uses a left-shift unit and a tree of 4-bit *OR*-gates to compute one point index at each clock cycle, assuming an *One-Node-per-Cycle* SD architecture.

The *dist_calc* module provides the predefined values of the metric $\delta^2_{\min}(\mathbf{s}_i)$, for the modified pruning metric computations. The pre-calculated values are stored in the *distance memory* ROM in ascending order. In the case of 16-QAM constellation there are 18 values of $\delta^2_{\min}(\mathbf{s}_i)$, while the memory requirements for the *distance memory* depend on the accuracy of the adopted bit representation. Since $\delta^2_{\min}(\mathbf{s}_i)$ is a lower limit of $\delta^2(\mathbf{s}_i)$, a reduced bit representation $\hat{\delta}^2_{\min}(\mathbf{s}_i)$ can be used to reduce the memory overhead as long as

$\hat{\delta}^2_{\min}(\mathbf{s}_i) \leq \delta^2_{\min}(\mathbf{s}_i)$, without compromising the decoding performance. The correct metric value, for a specific constellation point, is selected based on a Look-Up table (LUT), the *find distance memory* ROM. The address of the *find distance memory* is constructed by the concatenation of the index of $s^{(0)}$, the quadrant Q and the value of the *enumerated-counter*.

The *expanding_logic* unit computes the partial Euclidean distance (PED) metric $\Delta(s_i)$ for the specific constellation point and by adding the value of $D(\mathbf{s}_{i+1})$ compares the resulting partial distance with the sphere constraint (r^2) to perform vertical pruning. The *pruning_logic* module computes $\tilde{r}^2$, based on the values of r^2, $\frac{2\sigma_n^2}{|R_{i,i}|^2}$ and $D(\mathbf{s}_{i+1})$, and compares it with the $\delta^2_{\min}(\mathbf{s}_i)$ metric, of the enumerated constellation point, to perform horizontal and vertical pruning. In the proposed "approximate" version the *pruning_logic* and the *dist_calc* modules are not required. The horizontal and vertical pruning is performed by the *expanding_logic*.

The critical path delay[1] of the proposed "exact" and "approximate" implementations is shown in the Fig.6.9 with bold lines and includes: (a) the *find* $s^{(0)}$ *and* Q module, (b) the *vo_calc* unit, reading from (c) the *QAM ROM*, performing (d) the *PD-calculation*, (e) the addition $D(s_i) = \Delta(s_i) + D(s_{i+1})$ and (f) the comparison of the value of $D(s_i)$ with the r^2 value.

6.6 Algorithmic Performance Evaluation

This section presents a performance evaluation of the proposed enumeration technique, through several simulations. We consider a 4×4 MIMO-OFDM system, which is operating over a spatially and temporally uncorrelated Rayleigh flat-fading channel and the information bits are mapped onto a 16-QAM (Gray coding) constellation. Furthermore, a systematic $(5/7)_8$ convolutional code with rate 1/2 is used with a code block of 2304 information bits, while the log-MAP BCJR algorithm is used for channel decoding. Finally, the Single Tree Search (*STS*) approach of [287] is used to avoid multiple hard sphere decoding processes per bit, after modifications to support the proposed pruning metric instead of the exact calculated PD value.

As mentioned before we consider two versions of the proposed enumeration scheme.

[1] Critical path is the path with the maximum delay in the entire circuit.

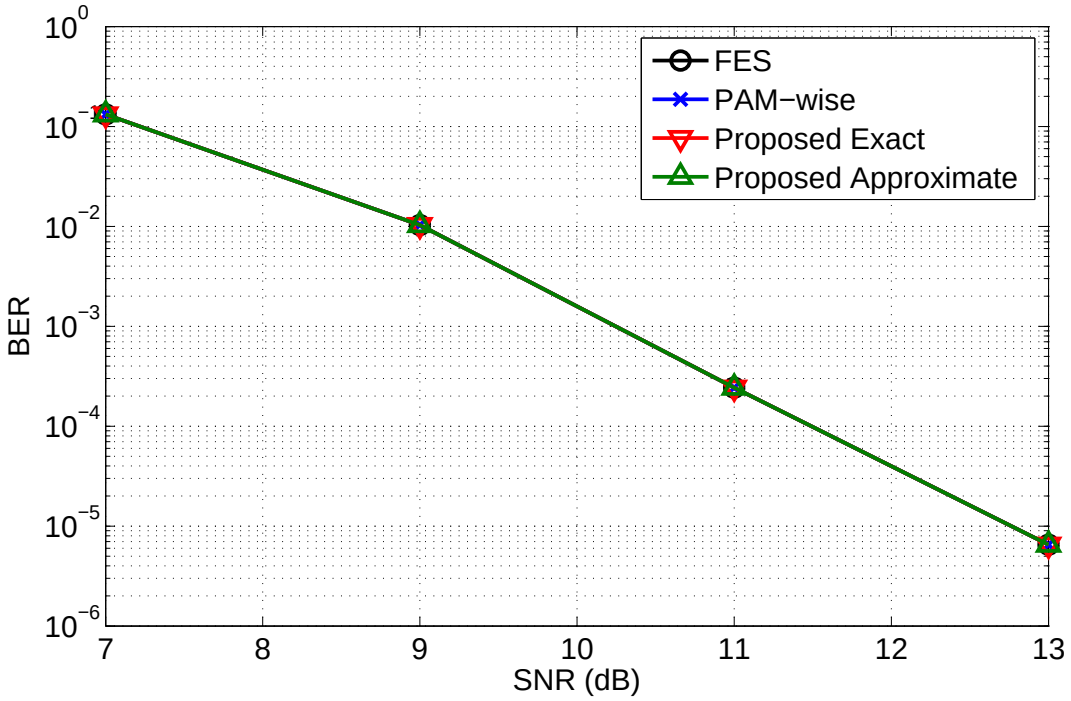


Figure 6.10 BER performance of the different enumeration schemes.

The "exact" version which uses the predefined ordering and the modified pruning metric and it can guarantee the *max-log MAP* performance of the detection process, and the "approximate" version which employs only the proposed visiting order while performing traditional pruning, with negligible performance loss. The evaluation of the novel enumeration technique is based on the comparison against the *Full Enumeration and Sort (FES)* and against the PAM-wise enumeration scheme, which groups the QAM points in PAM sub-constellations [99],[196], and performs SE enumeration (Section 6.2).

Figure 6.10 depicts the *Bit-Error Rate (BER)* performance of the four enumeration techniques. The "exact" version of the proposed enumeration has optimal decoding performance, like FES and PAM-wise technique. The "approximate" version of the proposed enumeration, while not optimal has a BER performance practically indistinguishable. This is because by using the Single Tree Search SD algorithm [287] at least one of the two minimization problems of (6.1) (the one with the smaller Euclidean distance) is always correctly calculated.

Furthermore, the use of a predefined visiting order based on a tight approximation of

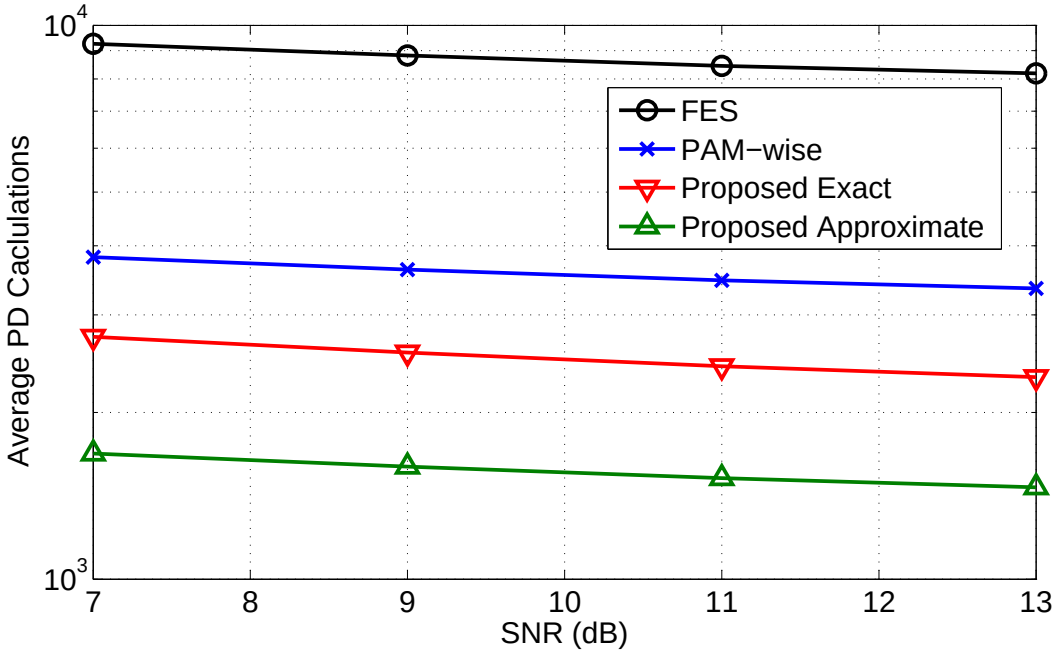


Figure 6.11 Average PD calculations for the different enumeration schemes.

the SE enumeration, provides a solution for the other minimization problem, very close to the actual one. Hence, the calculated LLR values, based on the "approximate" version of the enumeration, are accurate enough not to compromise the BER performance of the detection process. On the other hand, this near-optimal performance cannot be guaranteed for all soft-output sphere decoder implementations or soft-input soft-output SD approaches by using the techniques proposed in [216] and [166]. In contrast, the "exact" version of the proposed enumeration scheme can guarantee the optimal decoding for all operational scenarios and channel conditions.

The average required partial distance (PD) calculations and total number of visiting nodes of the proposed approaches are shown in figures 6.11 and 6.12, in comparison to those of the FES and PAM-wise enumeration techniques. In the simulations we assume that both pruning comparisons (i.e., $\delta^2_{\min}(\mathbf{s}_i)$ with $\tilde{r}^2$ and $D(\mathbf{s}_i)$ with r^2) are performed concurrently to minimize the critical path delay of the implementation (as shown in fig. 6.9), resulting to an equal number of PD calculations and visiting nodes. The less tight pruning metric of the proposed scheme results in increased number of visiting nodes (about 51%) but the reduced computational complexity per visited node leads to about 30% less PD cal-

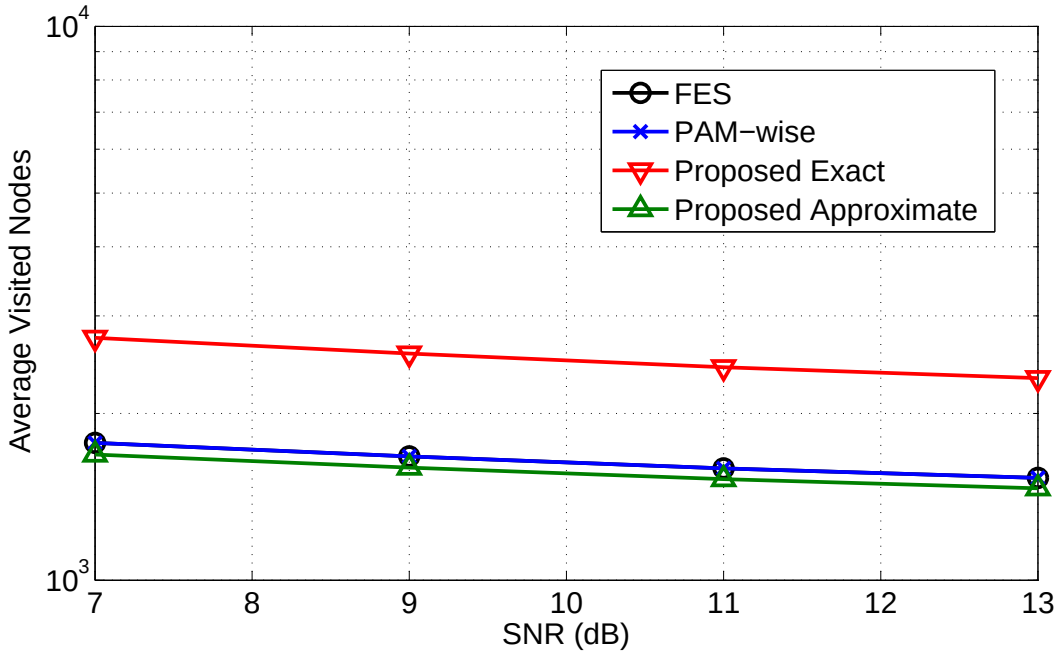


Figure 6.12 Average visited nodes for the different enumeration schemes.

culations compared to the PAM-wise enumeration technique, which performs more than one PD computation per node. The calculations of the $\tilde{r}^2$ values during the detection process, have negligible computational complexity (i.e. for 13 dB SNR an average of 24 calculation per SD is performed). We can reduce further the required PD calculations by performing the pruning comparisons serially, with the additional cost of increased critical path delay for the implementation. In this case the related gain will be 40% less PD calculations[2]. The "approximate" version of the proposed enumeration provide about 56% reduction in the total number of PD calculations, during the decoding process, with an additional 5% reduction in the number of visited nodes.

The implementation gains from the proposed scheme depend on the selected architecture of the sphere decoder. For architectures performing one PD calculation per clock cycle, the reduced number of PD computations leads to increased throughput of the detection process. In sphere decoder architecture similar to the "One-Node-Per-Cycle" of [287], the short critical path delay and the reduced PD calculations result in area and power/energy gains. We will discussed the second case in section 6.7.

[2]In simulations not shown here.

6.7 Implementation Comparison

This section presents comparisons for both the “exact” and “approximate” versions of the proposed enumeration scheme and the PAM-wise technique with respect to the hardware resources, performance and power consumption, for both FPGA and ASIC implementations. The PAM-wise implementation is similar to the PSK-wise approach [29], with one more PD calculation unit (for the case of 16-QAM) but low-complexity control circuit. The sorter critical path of the PAM-wise implementation, compared to that of the PSK-wise technique, results in more efficient implementation in terms of performance and it is beneficial in terms of a reference implementation, for benchmarking purposes.

The PAM-wise technique divides the 16-QAM constellation map into four PAM subsets (e.g. columns) and as mentioned in section 6.2, to minimize resources, a single ROM stores all these four subsets. Four *PD-calculation* units compute the PD metrics ($\Delta(s_i)$), and a tree of three comparators calculates the best candidate and performs the required comparison with r^2 that decides for node pruning, as shown in Fig.6.2. Hence, the PAM-wise critical path includes the *find CP and dir* module, the unit responsible to find the best candidate points for each subset, the *PD-calculation* module, the best candidate choice among the four, by the comparator tree of depth two, the addition $\Delta(s_i) + D(s_{i+1})$ and the comparison with r^2. The PAM-wise critical path is longer due to its tree of comparators compared to that of the proposed, which replaces the PAM-wise comparators tree with the *vo_calc* unit.

Performance and Hardware Resources

Tables 6.1 and 6.2 and Figures 6.13 and 6.14 show the FPGA resources utilization and the maximum operation frequency (performance) for both the “exact" and “approximate" proposed implementations, in comparison to the PAM-wise on different Xilinx FPGAs. The results are the final results after the Place and Route process. Three different FPGA models are used for each of the three Xilinx FPGA families (Virtex-6, Virtex-5 and Virtex-4), to ensure more accurate results and to avoid specific device optimizations. For a fair comparison, in terms of occupied hardware resources (FPGA slices) and maximum operational frequency, all implementations use the same fixed-point representations for the input/output data and for the intermediate results. Furthermore, the three implementations are mapped

without any FPGA optimizations, like the use of block RAMs and DSP blocks, and with normal effort for all processes (Synthesis, Map, Place & Route).

Table 6.1 FPGA Implementation Comparison (Resources utilization)

	PAM-wise	Proposed "Exact"		Proposed "Approximate"	
	Slices	Slices	Gain	Slices	Gain
XC6VLX240T-1	906	411	**54.64%**	285	**68.54%**
XC6VSX315T-1	911	429	**52.91%**	275	**69.81%**
XC6VHX250T-1	917	414	**54.85%**	284	**69.03%**
XC5VLX220T-1	777	365	**53.03%**	231	**70.27%**
XC5VSX240T-1	775	359	**53.67%**	253	**67.46%**
XC5VHX240T-1	772	378	**51.04%**	241	**68.78%**
XC4VLX200-10	1335	911	**31.76%**	553	**58.58%**
XC4VSX55-10	1348	936	**30.56%**	539	**60.02%**
XC4VFX140-10	1341	907	**32.36%**	561	**58.17%**

The hardware resources gains for the "exact" version of the proposed technique are on average 54.1%, 52.6% and 31.6% for the tree FPGA families, while the average gains for the "approximate" version are 69.1%, 68.8% and 58.9% respectively. The maximum operational frequency is improved in the "exact" version by 26.1%, 22.6% and 18.2% and in the "approximate" version by 25.6%, 22.3% and 17.8% in comparison with the PAM-wise implementation, for the tree FPGA families. While the "exact" and "approximate" version of the proposed implementation have the same critical path, as mentioned before, the small difference in the performance gains is due to the differentiations during placement and routing processes. Furthermore, the reduced gains in Virtex-4 FPGA devices are mainly related to the smaller slices in these FPGA devices and the lack of the merging optimizations of the distributed RAMs, ROMs and FPGA LUTs, which introduced in the Virtex-5 devices. Both the proposed architectures have more ROMs, compared to the PAM-wise enumeration technique, which are implemented as distributed FPGA LUTs and have a negative impact in both the hardware resources utilization and operational frequency gains.

The reduced hardware resources utilization of the proposed architectures, in comparison to the PAM-wise implementation, is mainly due to the reduced number of PD calculation units. The proposed enumeration algorithm needs only one PD computation module, while the PAM-wise enumeration scheme uses four units, one for each of the subsets. The partial

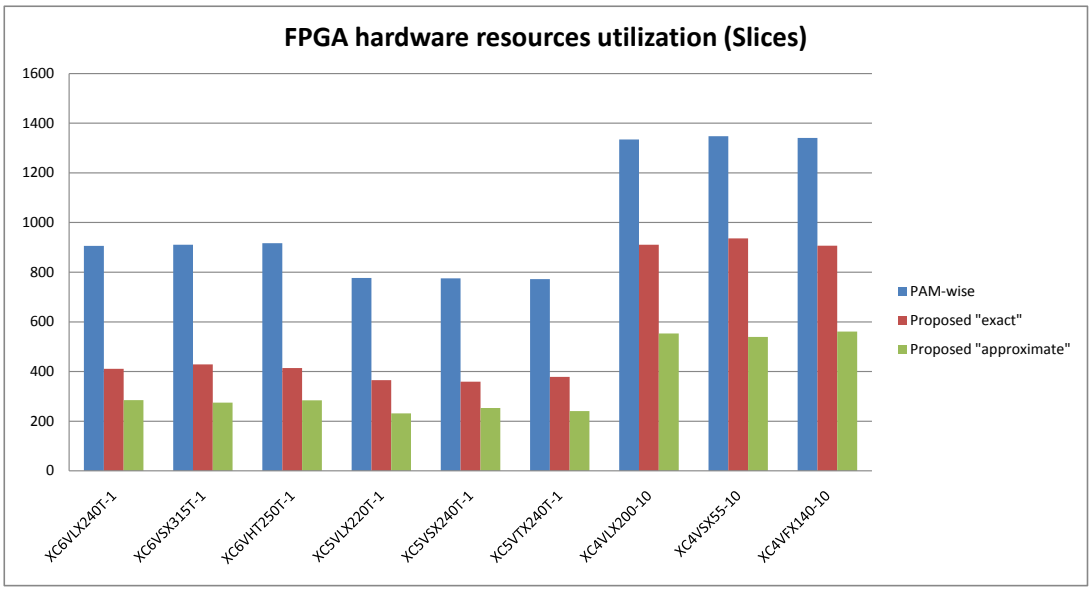


Figure 6.13 FPGA hardware resources utilization for PAE-wise, "exact" and "approximate" versions of the proposed implementations

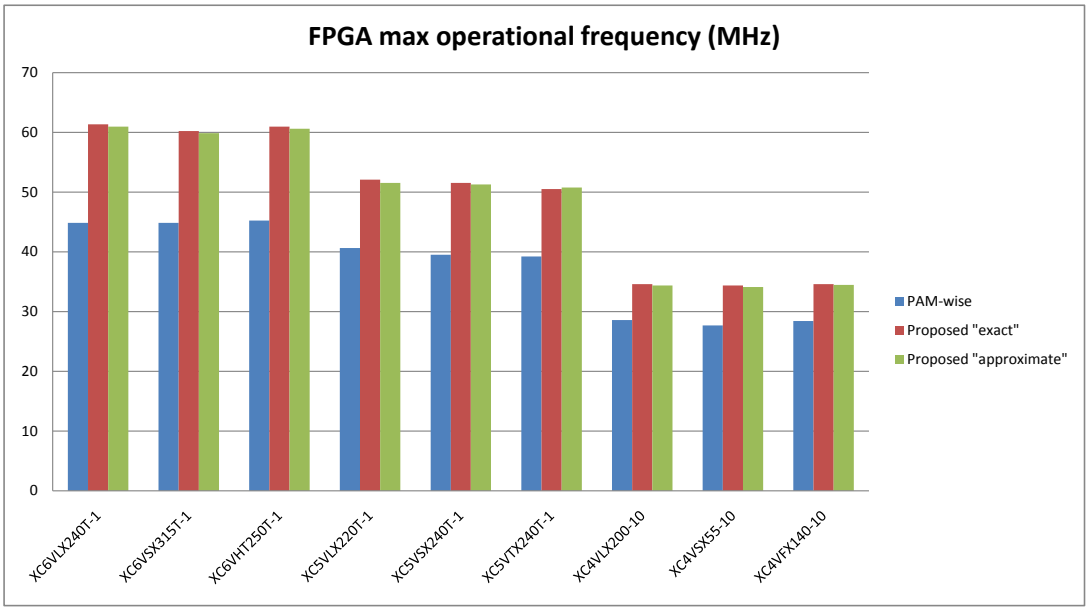


Figure 6.14 FPGA maximum operational frequency for PAE-wise, "exact" and "approximate" versions of the proposed implementations

Euclidean distance computation includes square multipliers which have a high hardware resource impact, in comparison with the other modules used in the three implementations. A more efficient, in terms of the total PD computation units requirements, is the PSK-wise enumeration scheme, which splits the 16-QAM constellation into three subset, in comparison

Table 6.2 FPGA Implementation Comparison (Performance)

	PAM-wise	Proposed "Exact"		Proposed "Approximate"	
	Freq (MHz)	Freq (MHz)	Gain	Freq (MHz)	Gain
XC6VLX240T-1	44.84	61.35	**26.91%**	60.98	**26.46%**
XC6VSX315T-1	44.84	60.24	**25.56%**	59.88	**25.11%**
XC6VHX250T-1	45.24	60.98	**25.79%**	60.61	**25.34%**
XC5VLX220T-1	40.65	52.08	**21.95%**	51.55	**21.14%**
XC5VSX240T-1	39.53	51.55	**23.32%**	51.28	**22.93%**
XC5VHX240T-1	39.22	50.51	**22.35%**	50.76	**22.75%**
XC4VLX200-10	28.57	34.60	**17.43%**	34.36	**16.86%**
XC4VSX55-10	27.70	34.36	**19.39%**	34.13	**18.84%**
XC4VFX140-10	28.41	34.60	**17.90%**	34.48	**17.61%**

to the four of the PAM-wise method, therefore it needs only three PD calculation modules, as mentioned in section 6.3. From Table 6.1 and Figure 6.13 we can see that the proposed approaches are more efficient, in terms of hardware resources utilization, even from the PSK-wise technique, without including the increased complexity of the control unit for the PSK-wise implementation.

Moreover, the proposed architectures prevail over the comparator based PAM-wise and PSK-wise implementations, in terms of critical path delay. The proposed approach is based on a predetermined visiting order, which requires only the sphere constraint comparator ($D(\mathbf{s}_i) \geq r^2$), in the critical path. The comparators tree of the PAM-wise and PSK-wise enumeration techniques increase the critical path delay of their implementations, in comparison to a simple candidate selection, from a LUT/ROM in the proposed architectures. As mentioned in section 6.3 the PSK-wise implementation has similar critical path delay with the PAM-wise architecture, with the additional complexity of the control unit.

Table 6.3 and Figure 6.15 show the comparison of the two proposed architectures and the PAM-wise technique, for the case of ASIC implementations, in terms of area and performance. The total area[3] is the post-synthesis results by using the Synopsys Design Compiler

[3]Both FPGA and ASIC implementation results include registers in the I/O ports of the design to ensure that the critical path delay measurements are accurate. The hardware resources utilization and the area results are evenly increased for all architectures due to the fact that all implementations have the same I/O ports.

and the TSMC 45nm standard-cell library in the typical corner[4].

Table 6.3 ASIC Implementation Comparison (Area & Performance)

Target Frequency (MHz)	PAM-wise	Proposed "Exact"		Proposed "Approximate"	
	Area (um^2)	Area (um^2)	Gain	Area (um^2)	Gain
100	7587.67	4819.95	**36.48%**	3146.80	**58.53%**
200	7587.49	4818.90	**36.49%**	3148.03	**58.51%**
250	7994.98	4818.90	**39.73%**	3146.62	**60.64%**
333	8635.84	4859.47	**43.73%**	3183.49	**63.14%**
400	8867.63	4867.58	**45.11%**	3287.21	**62.93%**
500	9590.34	5061.09	**47.23%**	3497.66	**63.53%**
556	9975.07	5211.39	**47.76%**	3575.45	**64.16%**
625	11079.86	5417.07	**51.11%**	3753.44	**66.12%**
667	12721.44	5603.70	**55.95%**	3884.50	**69.46%**
690	14046.38	5649.21	**59.78%**	4071.31	**71.02%**
714	–	5730.53	–	4214.90	–
769	–	6079.63	–	4849.94	–
833	–	7118.62	–	6049.29	–
862	–	7877.85	–	6783.24	–

As mentioned in Section 4.5, in the ASIC implementation flow the synthesis tool performs several optimization iterations for both targeting frequency and minimum occupied area. If the critical path of the design is relaxed, in the targeting frequency constraint, then more aggressive area optimization can be performed, otherwise less area can be saved. In the case of very strict clock constraints, in which the targeting frequency can not be achieved easily, aggressive speed optimizations should be performed by the synthesis tool, which can increase the area, due to register duplication, buffer insertion, etc.

The proposed architectures have more relaxed critical path, in comparison with the PAM-wise implementation, resulting in increased maximum frequency and increased area gains for more strict clock constraints. For targeting frequencies of 100 to 200 MHz, all architectures have relaxed critical paths with the proposed "exact" implementation having 36.49% area gain and proposed "approximate" having 58.53% gain, over the PAM-wise implementation. As targeting frequency of the ASIC synthesis is increasing, the proposed architectures, have a small area overhead, while the PAM-wise implementation increases the

[4]The typical corner for the TSMC 45nm std-cell library is 0.9V and 25°C.

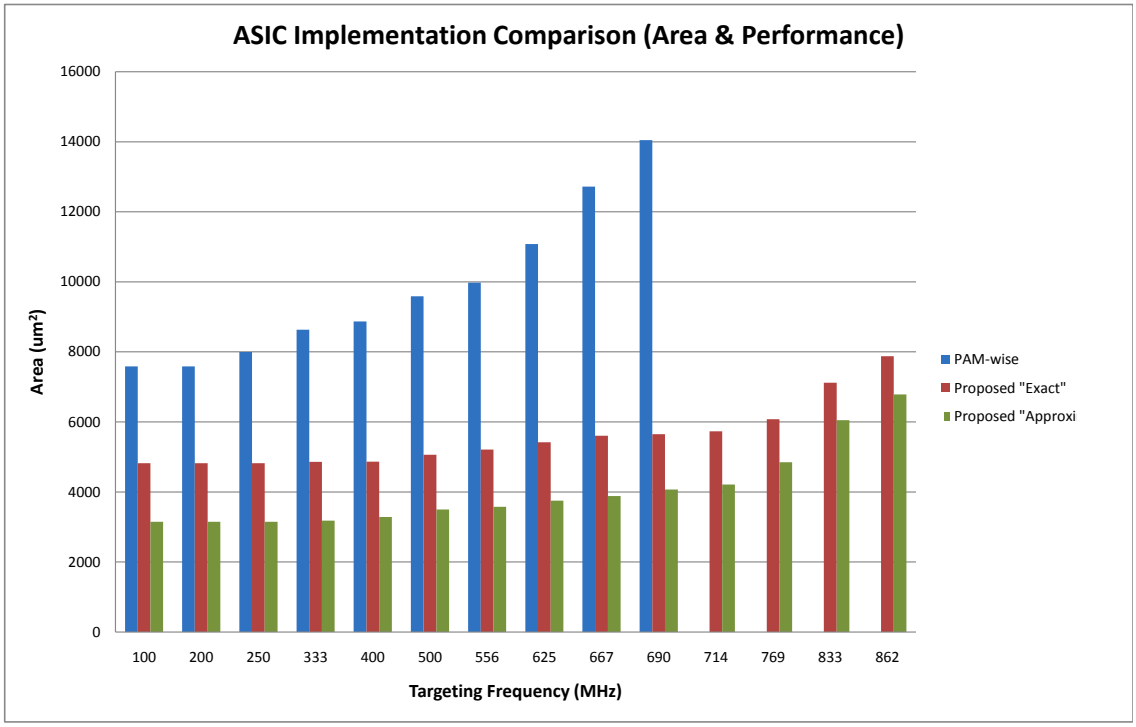


Figure 6.15 ASIC implementation comparison for PAE-wise, “exact” and “approximate” versions of the proposed implementations

occupied area with high rate, as illustrated in Fig. 6.15. For target frequencies of 556 MHz the area gains for the proposed “exact” and “approximate” implementations are increased to 47.76% and 64.16% respectively. The maximum frequency for the PAM-wise architecture is 690MHz and the area gains, at this clock constraint, of the proposed architectures are 59.78% and 71.02%. Finally, the maximum clock frequency for the proposed architecture is 862MHz, which is 25% higher than that of the PAM-wise implementation.

Power Consumption

Figure 6.16 show the total and the leakage power consumption of the PAM-wise and the proposed ASIC implementations, for different target clock frequencies. These power consumption results are reported by the Synopsys Design Compiler tool after a successful synthesis. The total power of a design implementation is the sum of the “static” or “leakage” power and the “dynamic” or “switching” power. The leakage power is the power consumed when the design have zero switching activity, while the dynamic power is the consumed power of the design having non-zero switching activity.

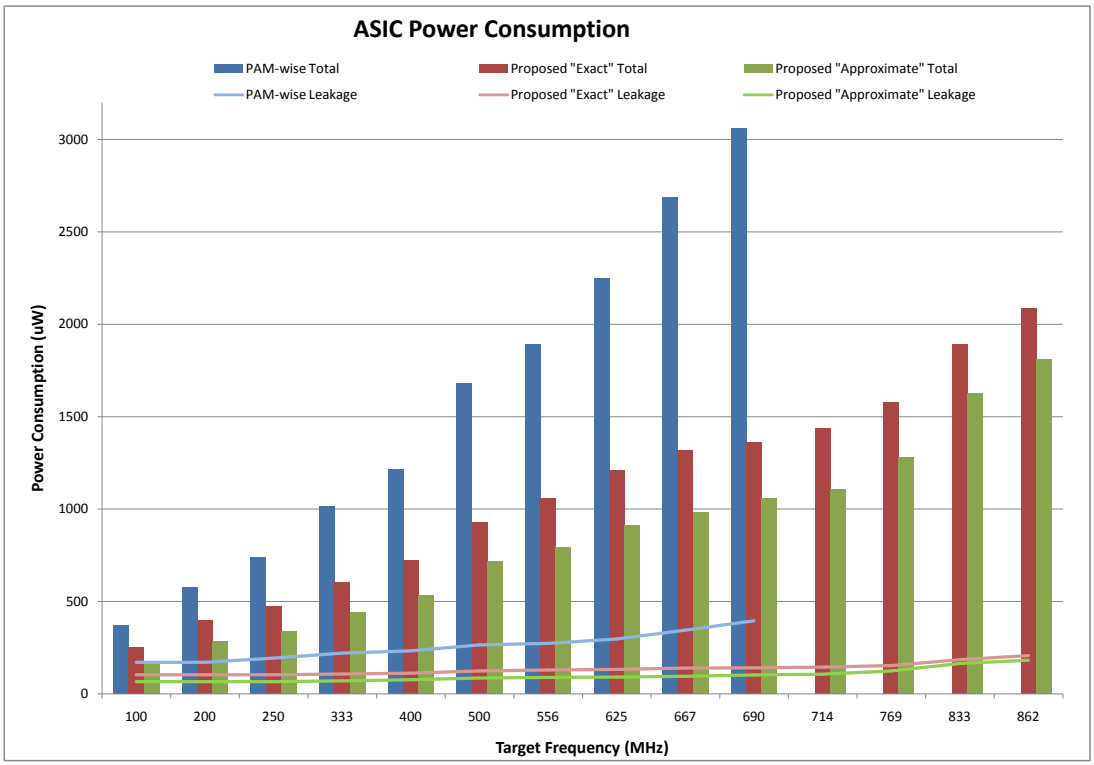


Figure 6.16 ASIC implementation total and leakage power consumption comparison for PAE-wise, "exact" and "approximate" versions of the proposed implementations

The synthesis tool calculates the leakage power consumption of the implementation, which depends on the total number and different types of the standard cells used for the specific design and the characteristics of each type of the cells. For the dynamic power consumption the tool is not aware of the circuit switching activity over the enumeration process, which depends on a number of parameters such as the sphere constraint, the channel conditions, etc. Without the use of several simulations of the designs with different parameters, it is very difficult to calculate the switching activity for the implementations. The synthesis tool can estimate the dynamic power consumed for a design by using a statistical model with a range of probabilities for each of the cells switching activity. In most of the cases, this estimation can be considered as the upper bound of the dynamic power consumption of the implementation.

No power optimization techniques are used, either for dynamic or leakage power reduction, in the three implementations, such as fine-grain clock-gating, specific power optimized cells selection etc. The proposed "exact" and "approximate" implementations consumes

Table 6.4 ASIC Implementation Power Consumption Comparison (Total Power)

Target Frequency (MHz)	PAM-wise	Proposed "Exact"		Proposed "Approximate"	
	Total (uW)	Total (uW)	Gain	Total (uW)	Gain
100	373.30	250.70	**32.84%**	175.20	**53.07%**
200	576.80	397.80	**31.03%**	284.60	**50.66%**
250	736.10	471.30	**35.97%**	339.20	**53.92%**
333	1013.20	605.40	**40.25%**	441.00	**56.47%**
400	1216.70	722.00	**40.66%**	533.20	**56.18%**
500	1681.70	925.40	**44.97%**	714.20	**57.53%**
556	1891.80	1058.80	**44.03%**	793.30	**58.07%**
625	2247.90	1209.00	**46.22%**	910.20	**59.51%**
667	2687.60	1319.80	**50.89%**	984.40	**63.37%**
690	3059.60	1362.40	**55.47%**	1055.90	**65.49%**
714	–	1436.30	–	1109.50	–
769	–	1577.90	–	1278.40	–
833	–	1889.10	–	1624.90	–
862	–	2087.50	–	1812.30	–

less total power than the PAM-wise architecture, as illustrated in Fig. 6.16 and table 6.4. The low-complexity implementations of the proposed architectures result in more efficient area optimizations from the synthesis tool and the use of more power efficient cells, due to relaxed critical paths. In the case of target clock frequencies of 100 to 200 MHz, for which all the implementations are relaxed, the proposed "exact" architecture has 32.84% power savings, while the "approximate" implementation has 53.07%. Both the proposed architectures increase the power consumption gains as the clock frequency constraint gets more strict, due to low-complexity implementation and relaxed critical paths. At the maximum clock frequency for the PAM-wise architecture the power savings are 55.45% and 65.49% for the "exact" and "approximate" versions, of the proposed technique. Furthermore, the proposed architectures, at the maximum clock frequency of 862 MHz, consumes about the same power as the PAM-wise implementation at a clock frequency of 556 MHz.

Table 6.5 shows the Leakage Power for the three implementations, in several target clock frequencies. The leakage power consumption is also shown in the Fig. 6.16 with solid lines. The increased leakage power consumption of the PAM-wise architecture, is a result of the increased number of cells used by the synthesizer, to implement a high complexity design,

Table 6.5 ASIC Implementation Power Consumption Comparison (Leakage Power)

Target Frequency (MHz)	PAM-wise	Proposed "Exact"		Proposed "Approximate"	
	Total (uW)	Total (uW)	Gain	Total (uW)	Gain
100	170.00	103.21	**39.29%**	65.77	**61.31%**
200	170.09	103.08	**39.40%**	65.56	**61.46%**
250	192.78	103.00	**46.57%**	65.78	**65.88%**
333	220.35	107.57	**51.18%**	70.03	**68.22%**
400	232.22	111.68	**51.91%**	76.59	**67.02%**
500	264.79	123.87	**53.22%**	85.90	**67.56%**
556	272.81	128.63	**53.22%**	85.90	**67.56%**
625	296.54	132.23	**55.41%**	90.86	**69.36%**
667	344.03	139.29	**59.51%**	94.75	**72.46%**
690	395.04	140.66	**64.39%**	102.07	**74.16%**
714	–	145.03	–	106.21	–
769	–	152.34	–	123.45	–
833	–	184.95	–	164.39	–
862	–	206.58	–	181.56	–

in comparison to the proposed enumeration technique. The gains on the leakage power consumption, of the proposed architectures are 39.29% and 61.31%, for the "exact" and "approximate" versions, for targeting clock frequencies on the range of 100 to 200 MHz, in which all the implementations are relaxed. The increasing leakage power consumption of the PAM-wise design, when the clock constraint gets more strict, is a result of the increased area and the use of more low-delay cells (which have more leakage power), to minimize the delay in the critical path. The leakage power consumption of the proposed architectures, is increasing with a very small rate, compared to the PAM-wise design, which is another evidence of a relaxed critical path and low-complexity implementation. Finally, the gains for the proposed designs, at the maximum clock frequency, for the PAM-wise implementation (690MHz), is 64.39% and 74.16%, respectively.

While the statistical modeling and estimation of the switching activity results, in most of the cases, to an upper bound of the actual dynamic power consumption, a simulation-based measurement of the switching activity or the power consumption, can reveal power dependencies on the system parameters, such as the sphere constraint (r^2), in our case.

By using the FPGA power analysis flow we measure the power consumption of the three architectures[5] in a Xilinx Virtex-6 device (XC6VLX240T-1), for different values of the sphere constraint.

As mentioned in the section 6.6 the "exact" version of the proposed enumeration scheme visits on average 51% more nodes, than the PAM-wise technique, due the less tight pruning metric. A more fair and accurate power consumption comparison requires that all the architectures should have the same average throughput (execution time) for the same sphere constraint value. Therefore, the proposed "exact" implementation operates at a frequency 51% higher than that of the PAM-wise architecture, while the proposed "approximate" design, which visits 5% fewer nodes than the PAM-wise enumeration, operates at a frequency which is reduced accordingly.

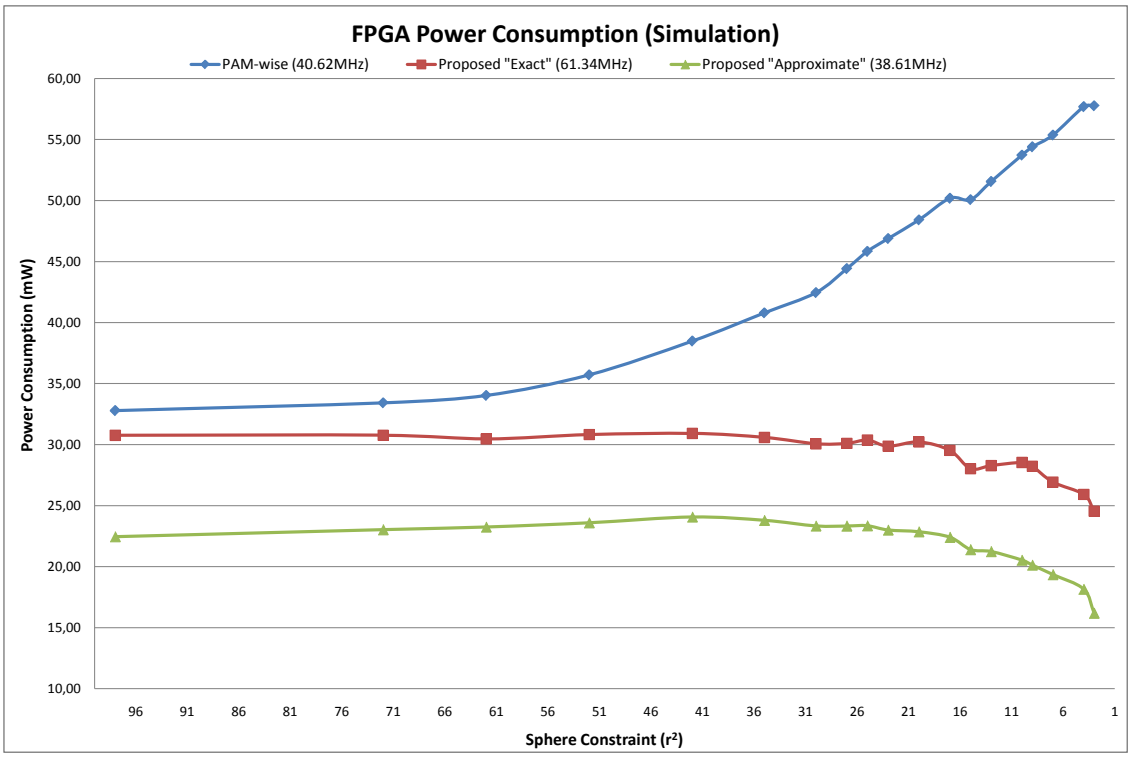


Figure 6.17 Power Consumption of PAM-wise and proposed ("exact" and "approximate") implementations.

Fig. 6.17 shows the power consumption of the three architectures, for several values of r^2. The proposed "exact" implementation, even though operates at a higher clock frequency

[5]While the FPGA implementations have registers in the I/O ports, the power measurements include only the designs activity and not the I/O registers power consumption.

(by 51%), has an average power consumption of 34% less than the PAM-wise implementation. Especially for small values of r^2 the improvement can be more than 57%. The "approximate" version of the proposed scheme has average power consumption 50% less than that of the PAM-wise enumeration and can reach an improvement of 68% for small sphere constraint values. The power savings of the proposed architectures are mainly due to the reduced number of PD calculations, for the same value of r^2, compared to the PAM-wise scheme, while the total number of comparison are reduced significantly. Furthermore, the low-complexity implementation and the shorter critical paths, of the proposed architectures have an additional impact in the reduction of the power consumption.

Fig. 6.17 also shows that the PAM-wise implementation has increased power consumption for smaller r^2, while the proposed implementation has almost constant power consumption. The four parallel PD metric computations of the PAM-wise's initialization step produce more switching activity and lead to increased power consumption. In normal operation the switching activity of the PAM-wise is due mainly to the comparator tree, the r^2 check and at most one of the PD calculation units. Therefore, for small r^2 the PAM-wise with more initializing steps has increased power consumption. Comparing the above performance to that of the proposed architectures we notice that there is no significantly difference in switching activity during the initialization and the normal operation.

The proposed implementation, based on predetermined visiting order needs at most one PD calculation per visiting node. Furthermore, the proposed technique performs less PD calculations regardless the value of the sphere constraint but the significant reduction is observed in the range of small values of r^2. For very large values of the sphere constraint the PD calculations performed by both schemes under comparison are the same. For small values of r^2 the proposed implementation shows reduced PD calculations (compared to the PAM-wise), greater than 55%, and reduced power consumption, as illustrated in Fig. 6.17. We should note that in practice, and especially with radius update, the sphere constraint becomes very small after calculating the first candidate solution, and therefore, the proposed scheme provides significant gains, over the PAM-wise enumeration. Finally, the difference between the power consumption of the "exact" and "approximate" versions of the proposed scheme, is due to the missing *pruning_logic* and *dist_calc* units from the proposed "approximate" implementation.

The proposed enumeration scheme reduces the complexity of the hard and soft-output sphere decoding algorithms, used to decode spatially multiplexed signals in MIMO-OFDM systems. As shown, in the previous sections the proposed approach is able to provide substantial complexity/implementation gains when high-order, non-constant envelop constellations (e.g 16-QAM) are transmitted, while it not compromises the *ML* or *max-log MAP* performance of the detection process. While the "exact" version of the proposed technique, increases the number of visiting nodes, during the tree-traversal, the reduced critical path of the implementation improves the requirements of hardware resources and power consumption, even with higher clock frequencies, in which the proposed architecture provides the same throughput compared to state-of-the-art approaches able to guarantee the optimal performance. Furthermore, reduced hardware resources are used for the implementation of the proposed technique, due to the predefined visiting order and the need for at most one PD computation per visiting node.

Chapter 7

Conclusions and Further Directions

The major challenge facing future wireless communication systems is to provide high data-rate and reliable wireless access to mobile users with increased quality-of-service (QoS) and efficient frequency spectrum utilization. Multiple-Input Multiple-Output (MIMO) wireless technology in combination with Orthogonal Frequency Division Multiplexing (OFDM) signals meets these demands by offering increased spectral efficiency, high data-rates and improved link reliability. On the other hand, these improvements come at the cost of significant increase in complexity of signal processing algorithms in both transmitter and receiver systems. The computational complexity of MIMO-OFDM receivers also increases dramatically with respect to the number of transmit/receive antennas as well as the modulation scheme used for transmission.

Furthermore, as the domain of mobile wireless communications becomes increasingly populated with differing communication protocols, the importance of mobile software defined radio (SDR) terminals grows. The high data-rates and multiple operational modes of new wireless protocols, in combination with the use of MIMO technologies, render an implementation on the limited processing resources of a flexible and scalable architecture extremely challenging. Moreover, the tight power consumption constraint, necessary to ensure long operation times from battery, does not relax this challenge either.

7.1 Conclusion

In the first part of this dissertation an efficient and fully scalable Macro-pipeline FFT architecture for concurrent multi-symbol processing on SDR/MIMO OFDM systems, has been presented. This reconfigurable memory-based FFT architecture is able to perform the FFT computations of multiple data streams with variable FFT lengths independently and can be tailored to match system requirements for the majority of multi-protocol (SDR) and/or MIMO OFDM systems. For single-protocol systems, the proposed architecture can be used for all the FFT calculation requirements of multiple data streams (MIMO mode), while supporting variable bandwidth configurations.

State of the art pipeline FFT architectures for multiple data streams are exploit the idle clock cycles of the butterfly processors for the computations of the additional data streams. High- and mixed-radix architectures are commonly used to support increased number of streams, while complex control and memory structures results in increased complexity and power consumption for the FFT processor. The limited scalability and the lack of support multiple streams with different FFT length each, makes these architectures unattractive for multi-protocol SDR systems, while the sequential output of the processed FFT frames, requires additional memory structures for the case of MIMO-OFDM systems.

Advanced memory-based FFT architectures have been used for multi-stream OFDM systems, combined with high- or mixed-radix butterfly processors and conflict-free addressing schemes. The majority of the state of the art memory-based architectures are based on a high-throughput FFT processor, which can be used for multiple data stream processing with a time scheduling scheme. The use of high- or mixed-radix processors increases the critical path of the butterfly unit and additional pipeline stages are required, while the complex addressing schemes or the cache structures used results in increased processing latency and memory requirements due to the non-normal order of the output data. These architectures support multiple data streams with different FFT lengths, but the limited scalability and the increased complexity for the support of continuous-flow operation, makes their implementation difficult for the majority of SDR/MIMO systems.

The proposed architecture contains multiple radix-2 butterfly processors which execute FFT calculations in parallel with each processor accessing data from two memory banks at each processing stage. The butterfly units can be configured at run-time to compute

smaller FFTs, by accessing their local memory banks, while a simple interconnection network allows each radix-2 processor to load (store) data from (to) the memory banks of other processors. The processing latency of the FFT processor can be optimized based on the configuration, the combination of the FFT lengths of the input data streams (SDR mode) and the run-time selection of the "local" FFT size for the radix-2 processors. A novel conflict-free in-place addressing scheme is used to minimize the processing latency for each radix-2 butterfly unit and to simplify the interconnection network between the processors and the memory banks. Furthermore, the proposed addressing scheme produces normal-order FFT outputs resulting in reduced memory requirements for a continuous-flow SDR/MIMO FFT architecture, in which the multiple FFT output frames can be accessed in parallel.

The FFT architecture, presented in Chapter 4, can be used in several SDR/MIMO-OFDM systems with different configurations in terms of number of data streams, FFT lengths and scheduling schemes. The novel conflict-free addressing scheme results in reduced complexity interconnection network and normal-order FFT output, while the run-time scheduling scheme reduces the processing latency by efficient utilize the butterfly processors, based on the FFT length of each of the data streams. The low implementation cost and the use of regular size memory structures is a key factor for reduced power consumption on large SDR/MIMO-OFDM systems, while the parallel normal-order FFT output results in low memory requirements for continuous-flow FFT processors.

An advanced LUT-based enumeration technique for sphere decoding algorithms, which is based on a predefined approximate visiting order and can guarantee the *ML* or *max-log MAP* performance of the MIMO detector, has been presented in the second part of this thesis. By exploiting geometrical characteristics of the constellation map, a pre-calculated visiting order can be defined and with appropriate tuning of the pruning metric the optimal solution can be guaranteed. While the proposed approach increases the number of visiting nodes, due to relaxed pruning metric, the low computational requirements for each node results in reduced total complexity and relaxed critical path for the sphere decoder.

Advanced enumeration schemes, which can guarantee the optimal solution, avoid the exhaustive search over all the constellation points, by splitting the QAM in several subsets. For each of the subsets a constellation point is selected, based on geometrical character-

istics, and the search for the next best candidate is limited to these selected constellation points. These approaches require several Euclidean distance calculation units and comparison modules, based on the total number of subsets, which increase the implementation cost and the power consumption of the MIMO detector. Other enumeration schemes perform a two dimensional search over the constellation points, while computing and store Euclidean distances, by using at most two distance calculation unit, independently of the QAM size. These techniques requires complex priority queues with several comparison modules resulting in increased implementation cost and power consumption.

Several enumeration schemes have been proposed in the literature, which reduce the computational complexity by using pre-determined approximate visiting order or by introducing approximations on the distance calculation process. These techniques can reduce the implementation cost and power consumption of the sphere decoder while introducing errors in the detection process resulting in performance degradation. While this performance loss can be negligible for specific scenarios, these approaches are unable to guarantee the optimal detection and hence increased error-rate performance, for all operational scenarios and channel conditions.

The proposed enumeration technique, while is based on a predefined visiting order can guarantee the optimal detection in all channel conditions, due to the tuning pruning metric. A single distance calculation unit is used in the enumeration process resulting in reduced implementation cost and very low power consumption, compared to other optimal enumeration schemes. It can be used for hard- and soft- output sphere decoders while small changes are required for the case of soft-input soft-output detection schemes. The "exact" and "approximate" versions of the proposed enumeration algorithm provide substantial complexity/implementation gains with high-order, non-constant envelop constellations, while having more relaxed critical path, which is a key factor for high-throughput sphere decoder implementations.

The increased number of visiting nodes, due to the more relaxed pruning metric, can be compensated with the low computational complexity for each node and hence reduced total complexity for the detection process. Furthermore, the relaxed critical path results in more efficient implementations with low power consumption, even in the cases of higher clock frequencies and increased detection throughput. Finally, the proposed "exact" architecture

results in very low power consumption, in comparison to state of the art optimal enumeration techniques, for the case of small sphere constraints, which is the common scenario in sphere decoder implementations using the radius update mechanism.

7.2 Future Work

The proposed FFT architecture for SDR/MIMO systems, supporting multiple data streams with variable symbol lengths, can be optimized further for the case of continuous-flow operation. The conflict-free addressing scheme produces normal-order output which is not similar to the bit-reversal input data ordering, resulting in low memory utilization of the I/O buffers. A tuned conflict-free addressing scheme can produce output data ordering similar to the input data, resulting in a single I/O buffer architecture with reduced memory requirements for the continuous-flow operation. Furthermore, a hybrid DIT/DIF scheme can be used with a similar conflict-free addressing technique to eliminate the use of multiple I/O buffers.

A mixed radix-2^2/2 could be used in combination with new conflict-free addressing techniques to further reduce the processing latency for large FFT sizes. The new butterfly unit will require a more complex interconnection network and a more efficient scheduling scheme, based on the system configuration and run-time parameters. The efficiency of the new conflict-free addressing scheme is a key factor for a reduced interconnection network and simplified control structures. Furthermore, low-power techniques can be used to further reduce the power consumption of the FFT processor for specific operational scenarios and system configurations.

The enumeration scheme, proposed in Chapter 6, has low implementation cost, due to single distance calculation unit, but it requires LUTs which are proportional to the constellation map. This can be a problem for very dense constellations which will have increased memory requirements. We can reduce the size of the LUTs by exploiting the constellation points symmetries and some geometrical characteristics of the QAM. For a specific constellation map, the points included to a subset, based on their Euclidean distance can be found by exploiting geometrical characteristics. We can translate the problem and use only the constellation points of the first quadrature to reduce the memory requirements for the

LUTs.

Compared to other optimal enumeration techniques, the proposed method reduce significantly the power consumption, specifically for small sphere constraints, which is the common case for sphere decoders which use the radius update algorithm. We can further reduce the power consumption by applying power optimization techniques, such as clock-gating, in the implementation of the proposed method. Furthermore, low-complexity optimal MIMO detectors, based on the soft-output or soft-input soft-output (SISO) sphere decoding algorithms could be consider. Applying the proposed power efficient enumeration scheme to a SISO sphere decoder can be beneficial in the research for reduced complexity and power consumption iterative receiver architectures.

Bibliography

[1] Abdulhamid, H., Abdel-Raheem, E., and Tepe, K. E. (2007). Channel tracking techniques for OFDM systems in wireless access vehicular environments. In *2007 9th International Symposium on Signal Processing and Its Applications*, pages 1–4.

[2] Adeva, E. P., Shah, M. A., Mennenga, B., and Fettweis, G. (2011). Vlsi architecture for soft-output tuple search sphere decoding. In *2011 IEEE Workshop on Signal Processing Systems (SiPS)*, pages 222–227.

[3] Agarwal, R. C. and Cooley, J. W. (1986). Fourier transform and convolution subroutines for the IBM 3090 vector facility. *IBM J. Res. Dev.*, 30(2):145–162.

[4] Ahmed, T., Garrido, M., and Gustafsson, O. (2011). A 512-point 8-parallel pipelined feedforward FFT for WPAN. In *2011 Conference Record of the Forty Fifth Asilomar Conference on Signals, Systems and Computers (ASILOMAR)*, pages 981–984.

[5] Alard, M. and Lassalle, R. (1987). Principles of modulation and channel coding for digital broadcasting for mobile receivers. *EBU Technical Review*, pages 168–190.

[6] Alastalo, A. T. and Kahola, M. (2003). Smart-antenna operation for indoor wireless local-area networks using OFDM. *IEEE Transactions on Wireless Communications*, 2(2):392–399.

[7] Alias, M. Y., Samingan, A. K., Chen, S., and Hanzo, L. (2003). Multiple antenna aided OFDM employing minimum bit error rate multiuser detection. *Electronics Letters*, 39(24):1769–1770.

[8] Arogyaswami, P., Rohit, N., and Dhananjay, G. (2003). *Introduction to Space-Time Wireless Communications*. Cambridge University Press.

[9] Baas, B. M. (1996). An energy-efficient single-chip FFT processor. In *VLSI Circuits, 1996. Digest of Technical Papers., 1996 Symposium on*, pages 164–165.

[10] Baas, B. M. (1999). A low-power, high-performance, 1024-point FFT processor. *IEEE Journal of Solid-State Circuits*, 34(3):380–387.

[11] Baek, J. H., Son, B. S., Jo, B. G., Sunwoo, M. H., and Oh, S. K. (2003). A continuous flow mixed-radix FFT architecture with an in-place algorithm. In *Circuits and Systems, 2003. ISCAS '03. Proceedings of the 2003 International Symposium on*, volume 2, pages II–133–II–136 vol.2.

[12] Bai, L. and Choi, J. (2012). *Low Complexity MIMO Detection*. Springer US, New York, NY, USA.

[13] Bailey, D. H. (1989). FFTs in external of hierarchical memory. In *Proceedings of the 1989 ACM/IEEE Conference on Supercomputing*, Supercomputing '89, pages 234–242, New York, NY, USA. ACM.

[14] Barbero, L. G., Ratnarajah, T., and Cowan, C. (2008). A low-complexity soft-mimo detector based on the fixed-complexity sphere decoder. In *2008 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 2669–2672.

[15] Barbero, L. G. and Thompson, J. S. (2006a). A fixed-complexity MIMO detector based on the complex sphere decoder. In *2006 IEEE 7th Workshop on Signal Processing Advances in Wireless Communications*, pages 1–5.

[16] Barbero, L. G. and Thompson, J. S. (2006b). Performance analysis of a fixed-complexity sphere decoder in high-dimensional MIMO systems. In *2006 IEEE International Conference on Acoustics Speech and Signal Processing Proceedings*, volume 4, pages IV–IV.

[17] Barbero, L. G. and Thompson, J. S. (2006c). Rapid prototyping of a fixed-throughput sphere decoder for MIMO systems. In *2006 IEEE International Conference on Communications*, volume 7, pages 3082–3087.

[18] Barbero, L. G. and Thompson, J. S. (2008a). Extending a fixed-complexity sphere decoder to obtain likelihood information for turbo-MIMO systems. *IEEE Transactions on Vehicular Technology*, 57(5):2804–2814.

[19] Barbero, L. G. and Thompson, J. S. (2008b). Fixing the complexity of the sphere decoder for MIMO detection. *IEEE Transactions on Wireless Communications*, 7(6):2131–2142.

[20] Barhumi, I., Leus, G., and Moonen, M. (2003). Optimal training design for MIMO OFDM systems in mobile wireless channels. *IEEE Transactions on Signal Processing*, 51(6):1615–1624.

[21] Bi, G. and Jones, E. V. (1989a). A pipelined FFT processor for word-sequential data. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 37(12):1982–1985.

[22] Bi, G. and Jones, E. V. (1989b). A pipelined FFT processor for word-sequential data. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 37(12):1982–1985.

[23] Biglieri, E., Calderbank, R., Constantinides, A., Goldsmith, A., Paulraj, A., and Poor, H. V. (2007). *MIMO Wireless Communications*. Cambridge University Press.

[24] Bingham, J. A. C. (1990). Multicarrier modulation for data transmission: an idea whose time has come. *IEEE Communications Magazine*, 28(5):5–14.

[25] Borlenghi, F., Witte, E. M., Ascheid, G., Meyr, H., and Burg, A. (2011). A 772mbit/s 8.81bit/nj 90nm cmos soft-input soft-output sphere decoder. In *IEEE Asian Solid-State Circuits Conference 2011*, pages 297–300.

[26] Bourdoux, A., Cappelle, H., and Dejonghe, A. (2011). Channel tracking for fast time-varying channels in IEEE802.11p systems. In *2011 IEEE Global Telecommunications Conference - GLOBECOM 2011*, pages 1–6.

[27] Brigham, E. O. (1988). *The Fast Fourier Transform and its Applications*. Prentice-Hall signal processing series.

[28] Brunel, L. and Boutros, J. (1999). Euclidean space lattice decoding for joint detection in CDMA systems. In *Proceedings of the 1999 IEEE Information Theory and Communications Workshop (Cat. No. 99EX253)*, pages 129–.

[29] Burg, A., Borgmann, M., Wenk, M., Zellweger, M., Fichtner, W., and Bolcskei, H. (2005). Vlsi implementation of MIMO detection using the sphere decoding algorithm. *Solid-State Circuits, IEEE Journal of*, 40(7):1566–1577.

[30] Burg, A., Borgmanr, M., Wenk, M., Studer, C., and Bolcskei, H. (2006a). Advanced receiver algorithms for mimo wireless communications. In *Proceedings of the Design Automation Test in Europe Conference*, volume 1, pages 6 pp.–.

[31] Burg, A., Haene, S., Perels, D., Luethi, P., Felber, N., and Fichtner, W. (2006b). Algorithm and vlsi architecture for linear MMSE detection in MIMO-OFDM systems. In *2006 IEEE International Symposium on Circuits and Systems*, pages 4 pp.–.

[32] Burg, A., Wenk, M., and Fichtner, W. (2006c). Vlsi implementation of pipelined sphere decoding with early termination. In *2006 14th European Signal Processing Conference*, pages 1–5.

[33] Cannizzaro, R. C., Banelli, P., and Leus, G. (2006). Adaptive channel estimation for OFDM systems with doppler spread. In *2006 IEEE 7th Workshop on Signal Processing Advances in Wireless Communications*, pages 1–5.

[34] Cao, Z., Tureli, U., and Yao, Y.-D. (2004). Deterministic multiuser carrier-frequency offset estimation for interleaved OFDMA uplink. *IEEE Transactions on Communications*, 52(9):1585–1594.

[35] Chan, T. S. (2007). *Handbook of Computer Networks: LANs, MANs, WANs, the Internet, and Global, Cellular, and Wireless Networks*, volume 2. John Wiley & Sons, Inc., Hoboken, NJ, USA.

[36] Chang, C.-K., Hung, C.-P., and Chen, S.-G. (2003). An efficient memory-based FFT architecture. In *Circuits and Systems, 2003. ISCAS '03. Proceedings of the 2003 International Symposium on*, volume 2, pages II–129–II–132 vol.2.

[37] Chang, R. and Gibby, R. (1968). A theoretical study of performance of an orthogonal multiplexing data transmission scheme. *IEEE Transactions on Communication Technology*, 16(4):529–540.

[38] Chang, R. W. (1966). Synthesis of band-limited orthogonal signals for multichannel data transmission. *The Bell System Technical Journal*, 45(10):1775–1796.

[39] Chang, R. Y. and Chung, W. H. (2012). Best-first tree search with probabilistic node ordering for MIMO detection: Generalization and performance-complexity tradeoff. *IEEE Transactions on Wireless Communications*, 11(2):780–789.

[40] Chang, Y. N. (2008). An efficient VLSI architecture for normal I/O order pipeline FFT design. *IEEE Transactions on Circuits and Systems II: Express Briefs*, 55(12):1234–1238.

[41] Chang, Y.-N. (2012). Design of an 8192-point sequential I/O FFT chip. *Lecture Notes in Engineering and Computer Science*.

[42] Chen, B. and Wang, H. (2002). Maximum likelihood estimation of OFDM carrier frequency offset. In *2002 IEEE International Conference on Communications. Conference Proceedings. ICC 2002 (Cat. No.02CH37333)*, volume 1, pages 49–53.

[43] Chen, R., Park, N., and Prasanna, V. K. (2013a). High throughput energy efficient parallel fft architecture on fpgas. In *High Performance Extreme Computing Conference (HPEC), 2013 IEEE*, pages 1–6.

[44] Chen, S., Zhang, T., and Xin, Y. (2007). Relaxed K-Best MIMO signal detector design and VLSI implementation. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 15(3):328–337.

[45] Chen, S.-C., Yu, C.-T., Tsai, C.-L., and Tang, J.-J. (2004). A new IFFT/FFT hardware implementation structure for OFDM applications. In *Circuits and Systems, 2004. Proceedings. The 2004 IEEE Asia-Pacific Conference on*, volume 2, pages 1093–1096 vol.2.

[46] Chen, S. G., Huang, S. J., Garrido, M., and Jou, S. J. (2014). Continuous-flow parallel bit-reversal circuit for MDF and MDC FFT architectures. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 61(10):2869–2877.

[47] Chen, X., He, G., and Ma, J. (2013b). Vlsi implementation of a high-throughput iterative fixed-complexity sphere decoder. *IEEE Transactions on Circuits and Systems II: Express Briefs*, 60(5):272–276.

[48] Chen, Y., Lin, Y. W., and Lee, C. Y. (2006). A block scaling FFT/IFFT processor for WiMAX applications. In *Solid-State Circuits Conference, 2006. ASSCC 2006. IEEE Asian*, pages 203–206.

[49] Chen, Y., Tsao, Y. C., Lin, Y. W., Lin, C. H., and Lee, C. Y. (2008). An indexed-scaling pipelined FFT processor for OFDM-based WPAN applications. *IEEE Transactions on Circuits and Systems II: Express Briefs*, 55(2):146–150.

[50] Chiueh, T.-D., Tsai, P.-Y., and Lai, I.-W. (2012). *Baseband Receiver Design for Wireless MIMO-OFDM Communications*. IEEE Press and John Wiley & Sons, Ltd.

[51] Cho, S. I., Kang, K. M., and Choi, S. S. (2008). Implemention of 128-point fast fourier transform processor for UWB systems. In *2008 International Wireless Communications and Mobile Computing Conference*, pages 210–213.

[52] Choi, J. (2006). Nulling and cancellation detector for MIMO and its application to multistage receiver for coded signals: performance and optimization. *IEEE Transactions on Wireless Communications*, 5(5):1207–1216.

[53] Choi, J., Yu, H., and Lee, Y. H. (2005). Adaptive MIMO decision feedback equalization for receivers with time-varying channels. *IEEE Transactions on Signal Processing*, 53(11):4295–4303.

[54] Chouliaras, V., Galiatsatos, P., Nakos, K., Reisis, D., and Vlassopoulos, N. (2009). Efficient cascaded VLSI FFT architecture for OFDM systems. In *Electronics, Circuits, and Systems, 2009. ICECS 2009. 16th IEEE International Conference on*, pages 97–100.

[55] Chu, E. and George, A. (1999). *Inside the FFT Black Box: Serial and Parallel Fast Fourier Transform Algorithms*. CRC Press, Computational Mathematics Series.

[56] Cimini, L. (1985). Analysis and simulation of a digital mobile channel using orthogonal frequency division multiplexing. *IEEE Transactions on Communications*, 33(7):665–675.

[57] Cirkic, M. and Larsson, E. G. (2014). Sumis: Near-optimal soft-in soft-out mimo detection with low and fixed complexity. *IEEE Transactions on Signal Processing*, 62(12):3084–3097.

[58] Clark, M. V. (1998). Adaptive frequency-domain equalization and diversity combining for broadband wireless communications. *IEEE Journal on Selected Areas in Communications*, 16(8):1385–1395.

[59] Cohen, D. (1976). Simplified control of FFT hardware. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 24(6):577–579.

[60] Coleri, S., Ergen, M., Puri, A., and Bahai, A. (2002). Channel estimation techniques based on pilot arrangement in OFDM systems. *IEEE Transactions on Broadcasting*, 48(3):223–229.

[61] Cortes, A., Velez, I., and Sevillano, J. F. (2009). Radix r^k FFTs: Matricial representation and SDC/SDF pipeline implementation. *IEEE Transactions on Signal Processing*, 57(7):2824–2839.

[62] Costa, E., Zhang, Y., and Totaro, A. (2007). Joint adaptive FDMA/SDMA in chunk-based OFDM systems. In *2007 IEEE International Conference on Communications*, pages 4658–4663.

[63] Dagres, I. and Polydoros, A. (2009). Decision-directed least-squares phase perturbation compensation in OFDM systems. *IEEE Transactions on Wireless Communications*, 8(9):4784–4796.

[64] Dai, X. (2005). Carrier frequency offset estimation for OFDM/SDMA systems using consecutive pilots. *IEE Proceedings - Communications*, 152(5):624–632.

[65] Damen, M. O., Gamal, H. E., and Caire, G. (2003). On maximum-likelihood detection and the search for the closest lattice point. *IEEE Transactions on Information Theory*, 49(10):2389–2402.

[66] Damen, O., Chkeif, A., and Belfiore, J. C. (2000). Lattice code decoder for space-time codes. *IEEE Communications Letters*, 4(5):161–163.

[67] David, T. and Viswanath, P. (2005). *Fundamentals of wireless communication*. Cambridge university press.

[68] de Jong, Y. L. C. and Willink, T. J. (2005). Iterative tree search detection for MIMO wireless systems. *IEEE Transactions on Communications*, 53(6):930–935.

[69] Delaruelle, A., Huisken, J., van Loon, J., and Welten, F. (1994). A channel demodulator IC for digital audio broadcasting. In *Custom Integrated Circuits Conference, 1994., Proceedings of the IEEE 1994*, pages 47–50.

[70] Doelz, M. L., Heald, E. T., and Martin, D. L. (1957). Binary data transmission techniques for linear systems. *Proceedings of the IRE*, 45(5):656–661.

[71] Dowler, A., Nix, A., and McGeehan, J. (2003). Data-derived iterative channel estimation with channel tracking for a mobile fourth generation wide area OFDM system. In *Global Telecommunications Conference, 2003. GLOBECOM '03. IEEE*, volume 2, pages 804–808 Vol.2.

[72] Fan, C. P., Lee, M. S., and Su, G. A. (2006). A low multiplier and multiplication costs 256-point FFT implementation with simplified radix-2^4 SDF architecture. In *APCCAS 2006 - 2006 IEEE Asia Pacific Conference on Circuits and Systems*, pages 1935–1938.

[73] Fincke, U. and Pohst, M. (1985). Improved methods for calculating vectors of short length in a lattice, including a complexity analysis. *Mathematics of Computation*, 44(170):463–471.

[74] Fischer, R. F. (2002). *Precoding and Signal Shaping for Digital Transmission*. John Wiley & Sons, Inc., New York, NY, USA.

[75] Foschini, G. J. (1996). Layered space-time architecture for wireless communication in a fading environment when using multi-element antennas. *Bell Labs Technical Journal*, 1(2):41–59.

[76] Fu, B. and Ampadu, P. (2009). An area efficient fft/ifft processor for mimo-ofdm wlan 802.11n. *Journal of Signal Processing Systems*, 56(1):59–68.

[77] Gan, Y. H., Ling, C., and Mow, W. H. (2009). Complex lattice reduction algorithm for low-complexity full-diversity MIMO detection. *IEEE Transactions on Signal Processing*, 57(7):2701–2710.

[78] Garrido, M., Grajal, J., Sanchez, M. A., and Gustafsson, O. (2013). Pipelined radix-2^k feedforward FFT architectures. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 21(1):23–32.

[79] Gault, S., Hachem, W., and Ciblat, P. (2006). Joint sampling clock offset and channel estimation for OFDM signals: Cramer-rao bound and algorithms. *IEEE Transactions on Signal Processing*, 54(5):1875–1885.

[80] Gentleman, W. M. and Sande, G. (1966). Fast fourier transforms: For fun and profit. In *Proceedings of the November 7-10, 1966, Fall Joint Computer Conference*, AFIPS '66 (Fall), pages 563–578, New York, NY, USA. ACM.

[81] Georgis, G., Nikitopoulos, K., and Jamieson, K. (2017). Geosphere: an exact depth-first sphere decoder architecture scalable to very dense constellations. *IEEE Access*, PP(99):1–1.

[82] Ghaderipoor, A. and Tellambura, C. (2008). A statistical pruning strategy for Schnorr-Euchner sphere decoding. *IEEE Communications Letters*, 12(2):121–123.

[83] Gifford, S., Bergstrom, C., and Chuprun, S. (2005). Adaptive and linear prediction channel tracking algorithms for mobile OFDM-MIMO applications. In *MILCOM 2005 - 2005 IEEE Military Communications Conference*, pages 1298–1302 Vol. 2.

[84] Gorokhov, A. and Linnartz, J. P. (2004). Robust OFDM receivers for dispersive time-varying channels: equalization and channel acquisition. *IEEE Transactions on Communications*, 52(4):572–583.

[85] Gowaikar, R. and Hassibi, B. (2007). Statistical pruning for near-maximum likelihood decoding. *Signal Processing, IEEE Transactions on*, 55(6):2661–2675.

[86] Groginsky, H. L. and Works, G. A. (1970). A pipeline fast fourier transform. *IEEE Transactions on Computers*, C-19(11):1015–1019.

[87] Guo, Z. and Nilsson, P. (2004). Reduced complexity schnorr-euchner decoding algorithms for mimo systems. *IEEE Communications Letters*, 8(5):286–288.

[88] Guo, Z. and Nilsson, P. (2006). Algorithm and implementation of the K-best sphere decoding for MIMO detection. *IEEE Journal on Selected Areas in Communications*, 24(3):491–503.

[89] Hagenauer, J., Offer, E., and Papke, L. (1996). Iterative decoding of binary block and convolutional codes. *IEEE Transactions on Information Theory*, 42(2):429–445.

[90] Han, W., Arslan, T., Erdogan, A. T., and Hasan, M. (2005). Multiplier-less based parallel-pipelined fft architectures for wireless communication applications. In *Proceedings. (ICASSP '05). IEEE International Conference on Acoustics, Speech, and Signal Processing, 2005.*, volume 5, pages v/45–v/48 Vol. 5.

[91] Hanzo, L. L., Münster, M., Choi, B., and Keller, T. (2003a). *OFDM and MC-CDMA for Broadband Multi-User Communications, WLANs and Broadcasting*. IEEE Press and John Wiley & Sons, Ltd.

[92] Hanzo, L. L., Yang, L.-L., and E-L. Kuan, K. Y. (2003b). *Single and Multi-Carrier DS-CDMA: Multi-User Detection, Space-Time Spreading, Synchronization, Networking and Standards*. IEEE Press and John Wiley & Sons, Ltd.

[93] Hassibi, B. and Vikalo, H. (2001). On the expected complexity of sphere decoding. In *Conference Record of Thirty-Fifth Asilomar Conference on Signals, Systems and Computers (Cat.No.01CH37256)*, volume 2, pages 1051–1055 vol.2.

[94] Hassibi, B. and Vikalo, H. (2005). On the sphere-decoding algorithm - expected complexity. *IEEE Transactions on Signal Processing*, 53(8):2806–2818.

[95] He, S. and Torkelson, M. (1996). A new approach to pipeline FFT processor. In *Parallel Processing Symposium, 1996., Proceedings of IPPS '96, The 10th International*, pages 766–770.

[96] He, S. and Torkelson, M. (1998a). Design and implementation of a 1024-point pipeline FFT processor. In *Custom Integrated Circuits Conference, 1998. Proceedings of the IEEE 1998*, pages 131–134.

[97] He, S. and Torkelson, M. (1998b). Designing pipeline FFT processor for OFDM (de)modulation. In *Signals, Systems, and Electronics, 1998. ISSSE 98. 1998 URSI International Symposium on*, pages 257–262.

[98] Heideman, M., Johnson, D., and Burrus, C. (1984). Gauss and the history of the fast fourier transform. *ASSP Magazine, IEEE*, 1(4):14–21.

[99] Hess, C., Wenk, M., Burg, A., Luethi, P., Studer, C., Felber, N., and Fichtner, W. (2007). Reduced-complexity MIMO detector with close-to ML error rate performance. In *Proceedings of the 17th ACM Great Lakes Symposium on VLSI*, GLSVLSI '07, pages 200–203, New York, NY, USA. ACM.

[100] Hidalgo, J. A., Lopez, J., Arguello, F., and Zapata, E. L. (1999). Area-efficient architecture for fast fourier transform. *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, 46(2):187–193.

[101] Hirosaki, B. (1980). An analysis of automatic equalizers for orthogonally multiplexed QAM systems. *IEEE Transactions on Communications*, 28(1):73–83.

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

[103] Hirosaki, B., Hasegawa, S., and Sabato, A. (1986). Advanced groupband data modem using orthogonally multiplexed QAM technique. *IEEE Transactions on Communications*, 34(6):587–592.

[104] Hlawatsch, F. and Matz, G. (2011). *Wireless Communications Over Rapidly Time-Varying Channels*. Elsevier Ltd.

[105] Hochwald, B. and ten Brink, S. (2003). Achieving near-capacity on a multiple-antenna channel. *Communications, IEEE Transactions on*, 51(3):389–399.

[106] Huang, S. J. and Chen, S. G. (2012). A high-throughput radix-16 fft processor with parallel and normal input/output ordering for ieee 802.15.3c systems. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 59(8):1752–1765.

[107] Huang, Z. Y. and Tsai, P. Y. (2011). Efficient implementation of QR decomposition for gigabit MIMO-OFDM systems. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 58(10):2531–2542.

[108] Hung, C.-C., Chiu, P.-L., and Huang, Y.-H. (2010). A variable-length fft processor for 4 x 4 mimo-ofdm systems. In *VLSI Design Automation and Test (VLSI-DAT), 2010 International Symposium on*, pages 156–159.

[109] Hung, C. L., Long, S. S., and Shiue, M. T. (2009). A low power and variable-length FFT processor design for flexible MIMO OFDM systems. In *2009 IEEE International Symposium on Circuits and Systems*, pages 705–708.

[110] Hwang, K. (1979). *Computer Arithmetic: Principles, Architecture, and Design*. John Wiley & Sons, Inc.

[111] Hwang, Y.-T. and Chen, W.-D. (2008). A low complexity complex QR factorization design for signal detection in MIMO OFDM systems. In *2008 IEEE International Symposium on Circuits and Systems*, pages 932–935.

[112] Jacobson, A. T., Truong, D. N., and Baas, B. M. (2009). The design of a reconfigurable continuous-flow mixed-radix fft processor. In *2009 IEEE International Symposium on Circuits and Systems*, pages 1133–1136.

[113] Jalden, J., Barbero, L. G., Ottersten, B., and Thompson, J. S. (2007). Full diversity detection in MIMO systems with a fixed-complexity sphere decoder. In *2007 IEEE International Conference on Acoustics, Speech and Signal Processing - ICASSP '07*, volume 3, pages III–49–III–52.

[114] Jalden, J., Barbero, L. G., Ottersten, B., and Thompson, J. S. (2009). The error probability of the fixed-complexity sphere decoder. *IEEE Transactions on Signal Processing*, 57(7):2711–2720.

[115] Jalden, J. and Ottersten, B. (2005a). On the complexity of sphere decoding in digital communications. *Signal Processing, IEEE Transactions on*, 53(4):1474–1484.

[116] Jalden, J. and Ottersten, B. (2005b). Parallel implementation of a soft output sphere decoder. In *Conference Record of the Thirty-Ninth Asilomar Conference onSignals, Systems and Computers, 2005.*, pages 581–585.

[117] James W. Cooley, J. W. T. (1965). An algorithm for the machine calculation of complex fourier series. *Mathematics of Computation*, 19(90):297–301.

[118] Jang, S., Yang, G., Lee, S., and Jung, Y. (2013). Area-efficient fft processor for mimo-ofdm based sdr systems. *IEICE Electronics Express*, 10(15):20130490–20130490.

[119] Jia, L., Gao, Y., and Tenhunen, H. (1999). A pipelined shared-memory architecture for fft processors. In *Circuits and Systems, 1999. 42nd Midwest Symposium on*, volume 2, pages 804–807 vol. 2.

[120] Jia, Y., Vithanage, C., Andrieu, C., and Piechocki, R. (2006). Probabilistic data association for symbol detection in MIMO systems. *Electronics Letters*, 42:38–40(2).

[121] Jiang, H., Luo, H., Tian, J., and Song, W. (2005). Design of an efficient FFT processor for OFDM systems. *IEEE Transactions on Consumer Electronics*, 51(4):1099–1103.

[122] Jiang, M. and Hanzo, L. (2007). Multiuser MIMO-OFDM for next-generation wireless systems. *Proceedings of the IEEE*, 95(7):1430–1469.

[123] Jiang, T., Song, L., and Zhang, Y. (2010). *Orthogonal Frequency Division Multiple Access Fundamentals and Applications*. CRC Press Taylor & Francis Group.

[124] Jiang, Y., Varanasi, M. K., and Li, J. (2011). Performance analysis of ZF and MMSE equalizers for MIMO systems: An in-depth study of the high SNR regime. *IEEE Transactions on Information Theory*, 57(4):2008–2026.

[125] Jo, B. G. and Sunwoo, M. H. (2005). New continuous-flow mixed-radix (CFMR) FFT processor using novel in-place strategy. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 52(5):911–919.

[126] Johansson, S., He, S., and Nilsson, P. (1999). Wordlength optimization of a pipelined FFT processor. In *Circuits and Systems, 1999. 42nd Midwest Symposium on*, volume 1, pages 501–503 vol. 1.

[127] Johnson, L. G. (1992). Conflict free memory addressing for dedicated FFT hardware. *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, 39(5):312–316.

[128] Johnston, J. A. (1983). Parallel pipeline fast fourier transformer. *Communications, Radar and Signal Processing, IEE Proceedings F*, 130(6):564–572.

[129] Jones, V. K. and Raleigh, G. C. (1998). Channel estimation for wireless OFDM systems. In *IEEE GLOBECOM 1998 (Cat. NO. 98CH36250)*, volume 2, pages 980–985 vol.2.

[130] Jung, Y., Yoon, H., and Kim, J. (2003). New efficient FFT algorithm and pipeline implementation results for OFDM/DMT applications. *IEEE Transactions on Consumer Electronics*, 49(1):14–20.

[131] Jung-Yeol, O. and Myoung-Seob, L. (2005). New radix-2 to the 4th power pipeline FFT processor. *IEICE transactions on electronics*, 88(8):1740–1746.

[132] Kalet, I. (1989). The multitone channel. *IEEE Transactions on Communications*, 37(2):119–124.

[133] Kang, B. and Kim, J. (2012). Low complexity multi-point 4-channel fft processor for ieee 802.11n mimo-ofdm wlan system. In *Green and Ubiquitous Technology (GUT), 2012 International Conference on*, pages 94–97.

[134] Karachalios, A., Nakos, K., Reisis, D., and Alnuweiri, H. (2009). A new FFT architecture for 4 x 4 MIMO-OFDMA systems with variable symbol lengths. In *Innovations in Information Technology, 2009. IIT '09. International Conference on*, pages 80–84.

[135] Kim, K. J., Yue, J., Iltis, R. A., and Gibson, J. D. (2005). A QRD-M/Kalman filter-based detection and channel estimation algorithm for MIMO-OFDM systems. *IEEE Transactions on Wireless Communications*, 4(2):710–721.

[136] Kim, S. I., Oh, H. S., and Choi, H. K. (2008). Mid-amble aided OFDM performance analysis in high mobility vehicular channel. In *2008 IEEE Intelligent Vehicles Symposium*, pages 751–754.

[137] Kim, T. H. (2014). Low-complexity sorted qr decomposition for mimo systems based on pairwise column symmetrization. *IEEE Transactions on Wireless Communications*, 13(3):1388–1396.

[138] Koffman, I. and Roman, V. (2002). Broadband wireless access solutions based on ofdm access in ieee 802.16. *IEEE Communications Magazine*, 40(4):96–103.

[139] Koutsoyannis, R., Milder, P., Berger, C., Glick, M., Hoe, J., and Puschel, M. (2012). Improving fixed-point accuracy of FFT cores in O-OFDM systems. In *Acoustics, Speech and Signal Processing (ICASSP), 2012 IEEE International Conference on*, pages 1585–1588.

[140] Kovalyov, I. P. (2004). *SDMA for Multipath Wireless Channels*. Springer-Verlag Berlin Heidelberg.

[141] Kuhn, V. (2006). *Wireless Communications over MIMO Channels: Applications to CDMA and Multiple Antenna Systems*. John Wiley and Sons, Inc.

[142] Kuo, J.-C., Wen, C.-H., Lin, C.-H., and Wu, A.-Y. A. (2003a). VLSI design of a variable-length FFT/IFFT processor for OFDM-based communication systems. *EURASIP Journal on Advances in Signal Processing*, 2003(13):1–11.

[143] Kuo, J.-C., Wen, C.-H., and Wu, A.-Y. (2003b). Implementation of a programmable 64 - 2048-point FFT/IFFT processor for OFDM-based communication systems. In *Circuits and Systems, 2003. ISCAS '03. Proceedings of the 2003 International Symposium on*, volume 2, pages II–121–II–124 vol.2.

[144] Lai, R. H., Chen, C. M., Ting, P. A., and Huang, Y. H. (2009). A modified sorted-qr decomposition algorithm for parallel processing in mimo detection. In *2009 IEEE International Symposium on Circuits and Systems*, pages 1405–1408.

[145] Land, I. and Hoeher, P. A. (2001). Using the mean reliability as a design and stopping criterion for turbo codes. In *Proceedings 2001 IEEE Information Theory Workshop (Cat. No.01EX494)*, pages 27–29.

[146] Laroia, R., Uppala, S., and Li, J. (2004). Designing a mobile broadband wireless access network. *IEEE Signal Processing Magazine*, 21(5):20–28.

[147] Larsson, E. G. (2009). MIMO detection methods: How they work [lecture notes]. *IEEE Signal Processing Magazine*, 26(3):91–95.

[148] Larsson, E. G. and Jalden, J. (2008). Fixed-complexity soft MIMO detection via partial marginalization. *IEEE Transactions on Signal Processing*, 56(8):3397–3407.

[149] Latsoudas, G. and Sidiropoulos, N. D. (2005). A hybrid probabilistic data association-sphere decoding detector for multiple-input-multiple-output systems. *IEEE Signal Processing Letters*, 12(4):309–312.

[150] Lee, H., Lee, B., and Lee, I. (2006a). Iterative detection and decoding with an improved V-BLAST for MIMO-OFDM systems. *IEEE Journal on Selected Areas in Communications*, 24(3):504–513.

[151] Lee, J., Lee, H., in Cho, S., and Choi, S.-S. (2006b). A high-speed, low-complexity radix-2^4 FFT processor for MB-OFDM UWB systems. In *2006 IEEE International Symposium on Circuits and Systems*, pages 4 pp.–.

[152] Lee, S., Jung, Y., and Kim, J. (2007). Low complexity pipeline FFT processor for MIMO-OFDM systems. *IEICE Electronics Express*, 4(23):750–754.

[153] Lee, S. and Park, S. C. (2007). Modified SDF architecture for mixed DIF/DIT FFT. In *2007 IEEE International Symposium on Circuits and Systems*, pages 2590–2593.

[154] Letzepis, N. and Grant, A. (2009). Bit error rate estimation for turbo decoding. *IEEE Transactions on Communications*, 57(3):585–590.

[155] Li, G., Zhang, X., Lei, S., Xiong, C., and Yang, D. (2012). An early termination-based improved algorithm for fixed-complexity sphere decoder. In *2012 IEEE Wireless Communications and Networking Conference (WCNC)*, pages 624–629.

[156] Li, M., Bougard, B., Lopez, E. E., Bourdoux, A., Novo, D., Perre, L. V. D., and Catthoor, F. (2008a). Selective spanning with fast enumeration: A near maximum-likelihood MIMO detector designed for parallel programmable baseband architectures. In *2008 IEEE International Conference on Communications*, pages 737–741.

[157] Li, M., Bougard, B., Novo, D., Thillo, W. V., der Perre, L. V., and Catthoor, F. (2008b). Adaptive SSFE near-ML MIMO detector with dynamic search range and 80-103Mbps flexible implementation. In *IEEE GLOBECOM 2008 - 2008 IEEE Global Telecommunications Conference*, pages 1–5.

[158] Li, N. and van der Meijs, N. P. (2009). A radix 2^2 based parallel pipeline FFT processor for MB-OFDM UWB system. In *2009 IEEE International SOC Conference (SOCC)*, pages 383–386.

[159] Li, P., de Lamare, R. C., and Fa, R. (2011). Multiple feedback successive interference cancellation detection for multiuser MIMO systems. *IEEE Transactions on Wireless Communications*, 10(8):2434–2439.

[160] Li, P. and Lamare, R. C. D. (2012). Adaptive decision-feedback detection with constellation constraints for MIMO systems. *IEEE Transactions on Vehicular Technology*, 61(2):853–859.

[161] Li, S., Xu, H., Fan, W., Chen, Y., and Zeng, X. (2010a). A 128/256-point pipeline fft/ifft processor for mimo ofdm system ieee 802.16e. In *Proceedings of 2010 IEEE International Symposium on Circuits and Systems*, pages 1488–1491.

[162] Li, S., Xu, H., Fan, W., Chen, Y., and Zeng, X. (2010b). A 128/256-point pipeline fft/ifft processor for mimo ofdm system ieee 802.16e. In *Proceedings of 2010 IEEE International Symposium on Circuits and Systems*, pages 1488–1491.

[163] Li, W. and Wanhammar, L. (1999). A pipeline FFT processor. In *Signal Processing Systems, 1999. SiPS 99. 1999 IEEE Workshop on*, pages 654–662.

[164] Li, Y. G., Winters, J. H., and Sollenberger, N. R. (2002). MIMO-OFDM for wireless communications: signal detection with enhanced channel estimation. *IEEE Transactions on Communications*, 50(9):1471–1477.

[165] Liang, Y., Schober, R., and Gerstacker, W. (2007). Transmit beamforming for frequency-selective channels with decision-feedback equalization. *IEEE Transactions on Wireless Communications*, 6(12):4401–4411.

[166] Liao, C.-H., Lai, I.-W., Nikitopoulos, K., Borlenghi, F., Kammler, D., Witte, M., Zhang, D., Chiueh, T.-D., Ascheid, G., and Meyr, H. (2009). Combining orthogonalized partial metrics: efficient enumeration for soft-input sphere decoder. In *Personal, Indoor and Mobile Radio Communications, 2009 IEEE 20th International Symposium on*, pages 1287–1291.

[167] Liao, C.-H., Wang, T.-P., and Chiueh, T.-D. (2010). A 74.8 mW soft-output detector IC for 8×8 spatial-multiplexing MIMO communications. *Solid-State Circuits, IEEE Journal of*, 45(2):411–421.

[168] Lin, D. D., Pacheco, R. A., Lim, T. J., and Hatzinakos, D. (2006). Joint estimation of channel response, frequency offset, and phase noise in OFDM. *IEEE Transactions on Signal Processing*, 54(9):3542–3554.

[169] Lin, H. L., Chang, R. C., and Chen, H. L. (2008). A high-speed SDM-MIMO decoder using efficient candidate searching for wireless communication. *IEEE Transactions on Circuits and Systems II: Express Briefs*, 55(3):289–293.

[170] Lin, L., Cimini, L. J., and Chuang, J. C. I. (2000). Comparison of convolutional and turbo codes for OFDM with antenna diversity in high-bit-rate wireless applications. *IEEE Communications Letters*, 4(9):277–279.

[171] Lin, Y., c. Liao, W., and y. Lee, C. (2005a). A MRMDF FFT processor for MIMO OFDM applications. In *2005 IEEE Asian Solid-State Circuits Conference*, pages 225–228.

[172] Lin, Y. W. and Lee, C. Y. (2007). Design of an FFT/IFFT processor for MIMO OFDM systems. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 54(4):807–815.

[173] Lin, Y.-W., Liu, H.-Y., and Lee, C.-Y. (2004). A dynamic scaling FFT processor for DVB-T applications. *IEEE Journal of Solid-State Circuits*, 39(11):2005–2013.

[174] Lin, Y.-W., Liu, H.-Y., and Lee, C.-Y. (2005b). A 1-GS/s FFT/IFFT processor for UWB applications. *IEEE Journal of Solid-State Circuits*, 40(8):1726–1735.

[175] Liu, H. and Lee, H. (2008). A high performance four-parallel 128/64-point radix-2^4 FFT/IFFT processor for MIMO-OFDM systems. In *Circuits and Systems, 2008. APCCAS 2008. IEEE Asia Pacific Conference on*, pages 834–837.

[176] Liu, L., Ren, J., Wang, X., and Ye, F. (2007). Design of low-power, 1GS/s throughput FFT processor for MIMO-OFDM UWB communication system. In *2007 IEEE International Symposium on Circuits and Systems*, pages 2594–2597.

[177] Liu, L., Ye, F., Ma, X., Zhang, T., and Ren, J. (2010). A 1.1-Gb/s 115-pJ/bit configurable MIMO detector using $0.13 - \mu$m CMOS technology. *Circuits and Systems II: Express Briefs, IEEE Transactions on*, 57(9):701–705.

[178] Liu, S.-Y. and Chong, J.-W. (2002). A study of joint tracking algorithms of carrier frequency offset and sampling clock offset for OFDM-based WLANs. In *IEEE 2002 International Conference on Communications, Circuits and Systems and West Sino Expositions*, volume 1, pages 109–113 vol.1.

[179] Liu, X., Yu, F., and Wang, Z.-k. (2011). A pipelined architecture for normal I/O order FFT. *Journal of Zhejiang University SCIENCE C*, 12(1):76–82.

[180] Lo, E. S., Chan, P. W. C., Lau, V. K. N., Cheng, R. S., Letaief, K. B., Murch, R. D., and Mow, W. H. (2007). Adaptive resource allocation and capacity comparison of downlink multiuser MIMO-MC-CDMA and MIMO-OFDMA. *IEEE Transactions on Wireless Communications*, 6(3):1083–1093.

[181] Lo, H.-F., Shieh, M.-D., and Wu, C.-M. (2001). Design of an efficient FFT processor for DAB system. In *Circuits and Systems, 2001. ISCAS 2001. The 2001 IEEE International Symposium on*, volume 4, pages 654–657 vol. 4.

[182] Loan, C. V. (1992). *Computational Frameworks for the Fast Fourier Transform*. Frontiers in Applied Mathematics, Society for Industrial and Applied Mathematics (SIAM).

[183] Luethi, P., Burg, A., Haene, S., Perels, D., Felber, N., and Fichtner, W. (2007). Vlsi implementation of a high-speed iterative sorted mmse qr decomposition. In *2007 IEEE International Symposium on Circuits and Systems*, pages 1421–1424.

[184] Luethi, P., Studer, C., Duetsch, S., Zgraggen, E., Kaeslin, H., Felber, N., and Fichtner, W. (2008). Gram-Schmidt-based QR decomposition for MIMO detection: VLSI implementation and comparison. In *APCCAS 2008 - 2008 IEEE Asia Pacific Conference on Circuits and Systems*, pages 830–833.

[185] Luo, J., Pattipati, K. R., Willett, P. K., and Hasegawa, F. (2001). Near-optimal multiuser detection in synchronous CDMA using probabilistic data association. *IEEE Communications Letters*, 5(9):361–363.

[186] Ma, L., Dickson, K., McAllister, J., and McCanny, J. (2011). QR decomposition-based matrix inversion for high performance embedded MIMO receivers. *IEEE Transactions on Signal Processing*, 59(4):1858–1867.

[187] Ma, W.-K., Davidson, T. N., Wong, K. M., Luo, Z.-Q., and Ching, P.-C. (2002). Quasi-maximum-likelihood multiuser detection using semi-definite relaxation with application to synchronous cdma. *IEEE Transactions on Signal Processing*, 50(4):912–922.

[188] Ma, X., Oh, M.-K., Giannakis, G. B., and Park, D.-J. (2005). Hopping pilots for estimation of frequency-offset and multiantenna channels in MIMO-OFDM. *IEEE Transactions on Communications*, 53(1):162–172.

[189] Ma, Y. (1999). An effective memory addressing scheme for FFT processors. *IEEE Transactions on Signal Processing*, 47(3):907–911.

[190] Ma, Y. and Wanhammar, L. (2000). A hardware efficient control of memory addressing for high-performance FFT processors. *IEEE Transactions on Signal Processing*, 48(3):917–921.

[191] Mahdavi, M. and Shabany, M. (2013). Novel MIMO detection algorithm for high-order constellations in the complex domain. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 21(5):834–847.

[192] Mansour, M. M., Alex, S. P., and Jalloul, L. M. A. (2014). Reduced complexity soft-output MIMO sphere detectors part i: Algorithmic optimizations. *IEEE Transactions on Signal Processing*, 62(21):5505–5520.

[193] Marsch, P., Zimmermann, E., and Fettweis, G. (2005). Smart candidate adding: A new low-complexity approach towards near-capacity MIMO detection. In *2005 13th European Signal Processing Conference*, pages 1–4.

[194] Mateer, T. D. (2008). The Fast Fourier Transform - A Mathematical Perspective. http://www.math.clemson.edu/∼janoski/reu/2008/FFT-book.pdf.

[195] May, T., Rohling, H., and Engels, V. (1998). Performance analysis of viterbi decoding for 64-DAPSK and 64-QAM modulated OFDM signals. *IEEE Transactions on Communications*, 46(2):182–190.

[196] Mennenga, B. and Fettweis, G. (2009). Search sequence determination for tree search based detection algorithms. In *Sarnoff Symposium, 2009. SARNOFF '09. IEEE*, pages 1–6.

[197] Mennenga, B., Matus, E., and Fettweis, G. (2009a). Vectorization of the sphere detection algorithm. In *2009 IEEE International Symposium on Circuits and Systems*, pages 2806–2809.

[198] Mennenga, B., von Borany, A., and Fettweis, G. (2009b). Complexity reduced soft-in soft-out sphere detection based on search tuples. In *2009 IEEE International Conference on Communications*, pages 1–6.

[199] Meyer, R. (1989). Error analysis and comparison of FFT implementation structures. In *Acoustics, Speech, and Signal Processing, 1989. ICASSP-89., 1989 International Conference on*, pages 888–891 vol.2.

[200] Milliner, D. L., Zimmermann, E., Barry, J. R., and Fettweis, G. (2008a). Channel state information based LLR clipping in list MIMO detection. In *2008 IEEE 19th International Symposium on Personal, Indoor and Mobile Radio Communications*, pages 1–5.

[201] Milliner, D. L., Zimmermann, E., Barry, J. R., and Fettweis, G. (2009). A fixed-complexity smart candidate adding algorithm for soft-output MIMO detection. *IEEE Journal of Selected Topics in Signal Processing*, 3(6):1016–1025.

[202] Milliner, D. L., Zimmermann, E., Barry, J. R., and Fettweis, G. P. (2008b). A framework for fixed complexity breadth-first mimo detection. In *2008 IEEE 10th International Symposium on Spread Spectrum Techniques and Applications*, pages 133–139.

[203] Mobasher, A., Taherzadeh, M., Sotirov, R., and Khandani, A. K. (2007). A near-maximum-likelihood decoding algorithm for MIMO systems based on semi-definite programming. *IEEE Transactions on Information Theory*, 53(11):3869–3886.

[204] Mody, A. N. and Stuber, G. L. (2001). Synchronization for MIMO OFDM systems. In *Global Telecommunications Conference, 2001. GLOBECOM '01. IEEE*, volume 1, pages 509–513 vol.1.

[205] Moose, P. H. (1994). A technique for orthogonal frequency division multiplexing frequency offset correction. *IEEE Transactions on Communications*, 42(10):2908–2914.

[206] Morelli, M. and Mengali, U. (2001). A comparison of pilot-aided channel estimation methods for OFDM systems. *IEEE Transactions on Signal Processing*, 49(12):3065–3073.

[207] Murugan, A. D., Gamal, H. E., Damen, M. O., and Caire, G. (2006). A unified framework for tree search decoding: rediscovering the sequential decoder. *IEEE Transactions on Information Theory*, 52(3):933–953.

[208] Myllyla, M., Antikainen, J., Juntti, M., and Cavallaro, J. R. (2007). The effect of LLR clipping to the complexity of list sphere detector algorithms. In *2007 Conference Record of the Forty-First Asilomar Conference on Signals, Systems and Computers*, pages 1559–1563.

[209] Nakos, K., Reisis, D., and Vlassopoulos, N. (2008). Addressing technique for parallel memory accessing in radix-2 FFT processors. In *Electronics, Circuits and Systems, 2008. ICECS 2008. 15th IEEE International Conference on*, pages 53–56.

[210] Nazar, G. L., Gimmler, C., and Wehn, N. (2010). Implementation comparisons of the QR decomposition forMIMO detection. In *Proceedings of the 23rd Symposium on Integrated Circuits and System Design*, SBCCI '10, pages 210–214, New York, NY, USA. ACM.

[211] Ngatched, T. M. N. and Takawira, F. (2001). Simple stopping criterion for turbo decoding. *Electronics Letters*, 37(22):1350–1351.

[212] Nguyen, V. D. and Kuchenbecker, H. P. (2001). Block interleaving for soft decision viterbi decoding in OFDM systems. In *IEEE 54th Vehicular Technology Conference. VTC Fall 2001. Proceedings (Cat. No.01CH37211)*, volume 1, pages 470–474 vol.1.

[213] Nikitopoulos, K. and Ascheid, G. (2012). Approximate mimo iterative processing with adjustable complexity requirements. *IEEE Transactions on Vehicular Technology*, 61(2):639–650.

[214] Nikitopoulos, K., Karachalios, A., and Reisis, D. (2014a). Exact max-log map soft-output sphere decoding via approximate schnorr-euchner enumeration. *Vehicular Technology, IEEE Transactions on*, PP(99):1–1.

[215] Nikitopoulos, K. and Polydoros, A. (2005). Phase-impairment effects and compensation algorithms for OFDM systems. *IEEE Transactions on Communications*, 53(4):698–707.

[216] Nikitopoulos, K., Zhang, D., Lai, I.-W., and Ascheid, G. (2010). Complexity-efficient enumeration techniques for soft-input, soft-output sphere decoding. *Communications Letters, IEEE*, 14(4):312–314.

[217] Nikitopoulos, K., Zhou, J., Congdon, B., and Jamieson, K. (2014b). Geosphere: Consistently turning MIMO capacity into throughput. *SIGCOMM Comput. Commun. Rev.*, 44(4):631–642.

[218] Niskanen, J., Janhunen, J., and Juntti, M. (2010). Selective spanning with fast enumeration detector implementation reaching LTE requirements. In *2010 18th European Signal Processing Conference*, pages 1379–1383.

[219] Nuaymi, L. (2007). *WiMAX: Technology for Broadband Wireless Access*. John Wiley and Sons, Inc.

[220] Oppenheim, A. and Weinstein, C. (1972). Effects of finite register length in digital filtering and the fast Fourier transform. *Proceedings of the IEEE*, 60(8):957–976.

[221] Oppenheim, A. V. (1988). *Advanced Topics in Signal Processing*. Prentice-Hall Signal Processing Series.

[222] Oppenheim, A. V., Ronald, W. S., and John, R. B. (1989). *Discrete-Time Signal Processing*. Prentice-Hall Signal Processing Series, 2nd edition.

[223] Oppenheim, A. V. and Schafer, R. W. (1975). *Digital Signal Processing*. Prentice-Hall.

[224] Parhami, B. (2000). *Computer Arithmetic: Algorithms and Hardware Designs*. Oxford University Press, Inc.

[225] Parhi, Keshab, K. (1999). *VLSI Digital Signal Processing Systems - Design and Implementation*. John Wiley and Sons, Inc.

[226] Park, J. S., Jung, H., and Prasanna, V. K. (2006). Efficient FPGA-based implementations of MIMO-OFDM physical layer. In *Proceedings of 2006 International Conference on Engineering of Reconfigurable Systems and Algorithms*, pages 273–279.

[227] Park, J. S. and Ogunfunmi, T. (2010). FPGA implementation of the MIMO-OFDM physical layer using single FFT multiplexing. In *Proceedings of 2010 IEEE International Symposium on Circuits and Systems*, pages 2682–2685.

[228] Parker, D.S., J. (1980). Notes on Shuffle/Exchange-Type Switching Networks. *Computers, IEEE Transactions on*, C-29(3):213–222.

[229] Patel, D., Shabany, M., and Gulak, P. G. (2009). A low-complexity high-speed QR decomposition implementation for MIMO receivers. In *2009 IEEE International Symposium on Circuits and Systems*, pages 33–36.

[230] Pease, M. C. (1969). Organization of large scale fourier processors. *J. ACM*, 16(3):474–482.

[231] Peled, A. and Ruiz, A. (1980). Frequency domain data transmission using reduced computational complexity algorithms. In *Acoustics, Speech, and Signal Processing, IEEE International Conference on ICASSP '80.*, volume 5, pages 964–967.

[232] Peng, F. and Ryan, W. E. (2006). Low-complexity soft demapper for OFDM fading channels with ICI. In *IEEE Wireless Communications and Networking Conference, 2006. WCNC 2006.*, volume 3, pages 1549–1554.

[233] Petrus, P., Sun, Q., Ng, S., Cho, J., Zhang, N., Breslin, D., Smith, M., McFarland, B., Sankaran, S., Thomson, J., Mosko, R., Chen, A., Lu, T., Wang, Y. H., Zhang, X., Nakahira, D., Li, Y., Subramanian, R., Venkataraman, A., Kumar, P., Swaminathan, S., Gilbert, J., Choi, W. J., and Ye, H. (2007). An integrated draft 802.11n compliant mimo baseband and mac processor. In *2007 IEEE International Solid-State Circuits Conference. Digest of Technical Papers*, pages 266–602.

[234] Pham, D., Pattipati, K. R., Willett, P. K., and Luo, J. (2004a). A generalized probabilistic data association detector for multiple antenna systems. *IEEE Communications Letters*, 8(4):205–207.

[235] Pham, D., Pattipati, K. R., Willett, P. K., and Luo, J. (2004b). An improved complex sphere decoder for V-BLAST systems. *IEEE Signal Processing Letters*, 11(9):748–751.

[236] Polychronakis, N., Reisis, D., Tsilis, E., and Zokas, I. (2012). Conflict free, parallel memory access for radix-2 fft processors. In *Electronics, Circuits and Systems (ICECS), 2012 19th IEEE International Conference on*, pages 973–976.

[237] Pomerleau, A., Buijs, H., and Fournier, M. (1977). A two-pass fixed point fast fourier transform error analysis. *Acoustics, Speech and Signal Processing, IEEE Transactions on*, 25(6):582–585.

[238] Prasad, N. and Varanasi, M. K. (2004). Analysis of decision feedback detection for MIMO rayleigh-fading channels and the optimization of power and rate allocations. *IEEE Transactions on Information Theory*, 50(6):1009–1025.

[239] Qian, D., Huang, M.-F., Ip, E., Huang, Y.-K., Shao, Y., Hu, J., and Wang, T. (2011). 101.7-tb/s (370×294-gb/s) pdm-128qam-ofdm transmission over 3×55-km ssmf using pilot-based phase noise mitigation. In *Optical Fiber Communication Conference/National Fiber Optic Engineers Conference 2011*, page PDPB5. Optical Society of America.

[240] Rabiner, L. R. and Gold, B. (1975). *Theory and application of digital signal processing*. Englewood Cliffs, N.J., Prentice-Hall, Inc.,.

[241] Radhouane, R., Liu, P., and Modlin, C. (2000). Minimizing the memory requirement for continuous flow FFT implementation: continuous flow mixed mode FFT (CFMM-FFT). In *Circuits and Systems, 2000. Proceedings. ISCAS 2000 Geneva. The 2000 IEEE International Symposium on*, volume 1, pages 116–119 vol.1.

[242] Raleigh, G. G. and Cioffi, J. M. (1996). Spatio-temporal coding for wireless communications. In *Global Telecommunications Conference, 1996. GLOBECOM '96. 'Communications: The Key to Global Prosperity*, volume 3, pages 1809–1814 vol.3.

[243] Raleigh, G. G. and Cioffi, J. M. (1998). Spatio-temporal coding for wireless communication. *IEEE Transactions on Communications*, 46(3):357–366.

[244] Raleigh, G. G. and Jones, V. K. (1998). Multivariate modulation and coding for wireless communication. In *IEEE GLOBECOM 1998 (Cat. NO. 98CH36250)*, volume 6, pages 3261–3269 vol.6.

[245] Raleigh, G. G. and Jones, V. K. (1999). Multivariate modulation and coding for wireless communication. *IEEE Journal on Selected Areas in Communications*, 17(5):851–866.

[246] Rao, K. R., Kim, D. N., and Hwang, J. J. (2010). *Fast Fourier Transform - Algorithms and Applications*. Signals and Communication Technology, Springer Netherlands.

[247] Reisis, D. and Vlassopoulos, N. (2006). Address generation techniques for conflict free parallel memory accessing in fft architectures. In *2006 13th IEEE International Conference on Electronics, Circuits and Systems*, pages 1188–1191.

[248] Rohling, M., May, T., Bruninghaus, K., and Grunheid, R. (1999). Broad-band OFDM radio transmission for multimedia applications. *Proceedings of the IEEE*, 87(10):1778–1789.

[249] Roy, R. H. (1997). Spatial division multiple access technology and its application to wireless communication systems. In *1997 IEEE 47th Vehicular Technology Conference. Technology in Motion*, volume 2, pages 730–734 vol.2.

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

[251] Sanchez, M. A., Garrido, M., Lopez-Vallejo, M., and Grajal, J. (2008). Implementing FFT-based digital channelized receivers on FPGA platforms. *IEEE Transactions on Aerospace and Electronic Systems*, 44(4):1567–1585.

[252] Saponara, S., Rovini, M., Fanucci, L., Karachalios, A., Lentaris, G., and Reisis, D. (2012). Design and comparison of FFT VLSI architectures for SoC telecom applications with different flexibility, speed and complexity trade-offs. *Circuits, Systems, and Signal Processing*, 31(2):627–649.

[253] Schafhuber, D., Matz, G., and Hlawatsch, F. (2003). Kalman tracking of time-varying channels in wireless MIMO-OFDM systems. In *The Thrity-Seventh Asilomar Conference on Signals, Systems Computers, 2003*, volume 2, pages 1261–1265 Vol.2.

[254] Schenk, A., Fischer, R. F. H., and Lampe, L. (2009a). A new stopping criterion for the sphere decoder in uwb impulse-radio multiple-symbol differential detection. In *2009 IEEE International Conference on Ultra-Wideband*, pages 606–611.

[255] Schenk, A., Fischer, R. F. H., and Lampe, L. (2009b). A stopping radius for the sphere decoder and its application to msdd of dpsk. *IEEE Communications Letters*, 13(7):465–467.

[256] Schenk, A. and Fischer, R. F. R. (2010). A stopping radius for the sphere decoder: Complexity reduction in multiple-symbol differential detection. In *2010 International ITG Conference on Source and Channel Coding (SCC)*, pages 1–6.

[257] Schenk, T. C. W. and van Zelst, A. (2003). Frequency synchronization for MIMO OFDM wireless LAN systems. In *2003 IEEE 58th Vehicular Technology Conference. VTC 2003-Fall (IEEE Cat. No.03CH37484)*, volume 2, pages 781–785 Vol.2.

[258] Schmidl, T. M. and Cox, D. C. (1997). Robust frequency and timing synchronization for OFDM. *IEEE Transactions on Communications*, 45(12):1613–1621.

[259] Schnorr, C. and Euchner, M. (1994). Lattice basis reduction: improved practical algorithms and solving subset sum problems. *Mathematical Programming*, 66(1-3):181–199.

[260] Schwoerer, L. and Zielinski, E. (2005). Optimized fft architecture for mimo applications. In *Proc. European Signal Processing Conference (EUSIPCO)*.

[261] Seethaler, D., Artes, H., and Hlawatsch, F. (2006). Dynamic nulling-and-canceling for efficient near-ML decoding of MIMO systems. *IEEE Transactions on Signal Processing*, 54(12):4741–4752.

[262] Shabany, M. (2009). *VLSI Implementation of Digital Signal Processing Algorithms for MIMO/SISO Systems*. PhD thesis, University of Toronto.

[263] Shabany, M. and Gulak, P. G. (2009). A 0.13μm CMOS 655Mb/s 4×4 64-QAM K-Best MIMO detector. In *2009 IEEE International Solid-State Circuits Conference - Digest of Technical Papers*, pages 256–257,257a.

[264] Shabany, M., Su, K., and Gulak, P. G. (2008). A pipelined scalable high-throughput implementation of a near-ML K-best complex lattice decoder. In *2008 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 3173–3176.

[265] Shah, M. A., Mennenga, B., and Fettweis, G. (2010). Iterative soft-in soft-out sphere detection for 3gpp lte. In *2010 IEEE 71st Vehicular Technology Conference*, pages 1–5.

[266] Shao, R. Y., Lin, S., and Fossorier, M. P. C. (1999). Two simple stopping criteria for turbo decoding. *IEEE Transactions on Communications*, 47(8):1117–1120.

[267] Shen, C. A. and Eltawil, A. M. (2010). An adaptive reduced complexity k-best decoding algorithm with early termination. In *2010 7th IEEE Consumer Communications and Networking Conference*, pages 1–5.

[268] Shen, C. A., Eltawil, A. M., Salama, K. N., and Mondal, S. (2012). A best-first soft/hard decision tree searching MIMO decoder for a 4×4 64-QAM system. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 20(8):1537–1541.

[269] Shi, L.-Z. and Guo, J.-M. (2016). Design of an 8-channel fft processor for ieee 802.11ac mimo-ofdm wlan system. *Circuits, Systems, and Signal Processing*, 35(10):3759–3769.

[270] Shieh, S. L., Chiu, R. D., Feng, S. L., and Chen, P. N. (2013). Low-complexity soft-output sphere decoding with modified repeated tree search strategy. *IEEE Communications Letters*, 17(1):51–54.

[271] Shim, B. and Kang, I. (2008). Sphere decoding with a probabilistic tree pruning. *Signal Processing, IEEE Transactions on*, 56(10):4867–4878.

[272] Shin, C., Heath, R. W., and Powers, E. J. (2007). Blind channel estimation for MIMO-OFDM systems. *IEEE Transactions on Vehicular Technology*, 56(2):670–685.

[273] Shin, M. and Lee, H. (2008). A high-speed four-parallel radix-2^4 FFT/IFFT processor for UWB applications. In *2008 IEEE International Symposium on Circuits and Systems*, pages 960–963.

[274] Sibille, A., Oestges, C., and Zanella, A. (2010). *MIMO: From Theory to Implementation*. Academic Press, 1st edition.

[275] Sidiropoulos, N. D. and Luo, Z. Q. (2006). A semidefinite relaxation approach to MIMO detection for high-order QAM constellations. *IEEE Signal Processing Letters*, 13(9):525–528.

[276] Simeone, O., Bar-Ness, Y., and Spagnolini, U. (2004). Pilot-based channel estimation for OFDM systems by tracking the delay-subspace. *IEEE Transactions on Wireless Communications*, 3(1):315–325.

[277] Simon, E. P., Ros, L., Hijazi, H., Fang, J., Gaillot, D. P., and Berbineau, M. (2011). Joint carrier frequency offset and fast time-varying channel estimation for MIMO-OFDM systems. *IEEE Transactions on Vehicular Technology*, 60(3):955–965.

[278] Son, B. S., Jo, B. G., Sunwoo, M. H., and Kim, Y. S. (2002). A high-speed FFT processor for OFDM systems. In *Circuits and Systems, 2002. ISCAS 2002. IEEE International Symposium on*, volume 3, pages III–281–III–284 vol.3.

[279] Song, W.-G. and Lim, J.-T. (2006). Channel estimation and signal detection for MIMO-OFDM with time varying channels. *IEEE Communications Letters*, 10(7):540–542.

[280] Spencer, Q. H., Swindlehurst, A. L., and Haardt, M. (2004). Zero-forcing methods for downlink spatial multiplexing in multiuser MIMO channels. *IEEE Transactions on Signal Processing*, 52(2):461–471.

[281] Steingrimsson, B., Luo, Z.-Q., and Wong, K. M. (2003). Soft quasi-maximum-likelihood detection for multiple-antenna wireless channels. *IEEE Transactions on Signal Processing*, 51(11):2710–2719.

[282] Storn, R. (1994). Radix-2 FFT-pipeline architecture with reduced noise-to-signal ratio. *Vision, Image and Signal Processing, IEE Proceedings* -, 141(2):81–86.

[283] Strinati, E. C., Simoens, S., and Boutros, J. (2007). Error rate estimation based on soft output decoding and its application to turbo coding. In *2007 IEEE Wireless Communications and Networking Conference*, pages 72–76.

[284] Stuber, G. L., Barry, J. R., McLaughlin, S. W., Li, Y., Ingram, M. A., and Pratt, T. G. (2004). Broadband mimo-ofdm wireless communications. *Proceedings of the IEEE*, 92(2):271–294.

[285] Studer, C. and Bolcskei, H. (2008). Soft-input soft-output sphere decoding. In *2008 IEEE International Symposium on Information Theory*, pages 2007–2011.

[286] Studer, C. and Bolcskei, H. (2010). Soft-input soft-output single tree-search sphere decoding. *IEEE Transactions on Information Theory*, 56(10):4827–4842.

[287] Studer, C., Burg, A., and Bolcskei, H. (2008). Soft-output sphere decoding: algorithms and VLSI implementation. *Selected Areas in Communications, IEEE Journal on*, 26(2):290–300.

[288] Swarztrauber, P. N. (1987). Multiprocessor ffts. *Parallel Computing*, 5(1):197 – 210.

[289] Takahashi, D. (2001). *A Blocking Algorithm for FFT on Cache-Based Processors*, pages 551–554. Springer Berlin Heidelberg, Berlin, Heidelberg.

[290] Takala, J. H., Jarvinen, T. S., and Sorokin, H. T. (2003). Conflict-free parallel memory access scheme for FFT processors. In *Circuits and Systems, 2003. ISCAS '03. Proceedings of the 2003 International Symposium on*, volume 4, pages IV–524–IV–527 vol.4.

[291] Tan, P. H. and Rasmussen, L. K. (2001). The application of semidefinite programming for detection in CDMA. *IEEE Journal on Selected Areas in Communications*, 19(8):1442–1449.

[292] Tan, P. H. and Rasmussen, L. K. (2004). Multiuser detection in CDMA - a comparison of relaxations, exact, and heuristic search methods. *IEEE Transactions on Wireless Communications*, 3(5):1802–1809.

[293] Tang, S. N., Liao, C. H., and Chang, T. Y. (2012). An area- and energy-efficient multimode FFT processor for WPAN/WLAN/WMAN systems. *IEEE Journal of Solid-State Circuits*, 47(6):1419–1435.

[294] Tang, S. N., Tsai, J. W., and Chang, T. Y. (2010). A 2.4-GS/s FFT processor for OFDM-based WPAN applications. *IEEE Transactions on Circuits and Systems II: Express Briefs*, 57(6):451–455.

[295] Tellado, J., Hoo, L. M. C., and Cioffi, J. M. (2003). Maximum-likelihood detection of nonlinearly distorted multicarrier symbols by iterative decoding. *IEEE Transactions on Communications*, 51(2):218–228.

[296] Thoen, S., Deneire, L., der Perre, L. V., Engels, M., and Man, H. D. (2003). Constrained least squares detector for OFDM/SDMA-based wireless networks. *IEEE Transactions on Wireless Communications*, 2(1):129–140.

[297] Thoen, S., der Perre, L. V., Engels, M., and Man, H. D. (2002). Adaptive loading for OFDM/SDMA-based wireless networks. *IEEE Transactions on Communications*, 50(11):1798–1810.

[298] Thompson, C. (1983). Fourier transforms in VLSI. *Computers, IEEE Transactions on*, C-32(11):1047–1057.

[299] Tosato, F. and Bisaglia, P. (2002). Simplified soft-output demapper for binary interleaved COFDM with application to HIPERLAN/2. In *2002 IEEE International Conference on Communications. Conference Proceedings. ICC 2002 (Cat. No.02CH37333)*, volume 2, pages 664–668 vol.2.

[300] Tsai, P.-Y., Chen, C.-W., and Huang, M.-Y. (2011). Automatic ip generation of fft/ifft processors with word-length optimization for mimo-ofdm systems. *EURASIP J. Adv. Signal Process*, 2011:1:1–1:15.

[301] Tsai, P. Y. and Lin, C. Y. (2011). A generalized conflict-free memory addressing scheme for continuous-flow parallel-processing fft processors with rescheduling. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 19(12):2290–2302.

[302] Tsoulos, G. (2006). *MIMO System Technology for Wireless Communications*. CRC Press.

[303] van de Beek, J. J., Edfors, O., Sandell, M., Wilson, S. K., and Borjesson, P. O. (1995). On channel estimation in OFDM systems. In *1995 IEEE 45th Vehicular Technology Conference. Countdown to the Wireless Twenty-First Century*, volume 2, pages 815–819 vol.2.

[304] Vandenameele, P. (2001). *Space Division Multiple Access for Wireless Local Area Networks*. Kluwer Academic Publishers, Norwell, MA, USA.

[305] Vandenameele, P., Perre, L. V. D., and Engels, M. (2002). *Space Division Multiple Access for Wireless Local Area Networks*. Springer US.

[306] Vandenameele, P., Perre, L. V. D., Engels, M., Gyselinckx, B., and Man, H. D. (1999). A novel class of uplink of OFDM/SDMA algorithms: a statistical performance analysis. In *Gateway to 21st Century Communications Village. VTC 1999-Fall. IEEE VTS 50th Vehicular Technology Conference (Cat. No.99CH36324)*, volume 1, pages 324–328 vol.1.

[307] Vandenameele, P., Perre, L. V. D., Engels, M. G. E., Gyselinckx, B., and Man, H. J. D. (2000). A combined OFDM/SDMA approach. *IEEE Journal on Selected Areas in Communications*, 18(11):2312–2321.

[308] Veithen, D., Spruyt, P., Pollet, T., Peeters, M., Braet, S., de Wiel, O. V., and Weghe, H. V. D. (1999). A 70 Mb/s variable-rate DMT-based modem for VDSL. In *Solid-State Circuits Conference, 1999. Digest of Technical Papers. ISSCC. 1999 IEEE International*, pages 248–249.

[309] Verdu, S. (1998). *Multiuser Detection*. Cambridge University Press, New York, NY, USA, 1st edition.

[310] Viterbi, A. J. (1995). *CDMA: Principles of Spread Spectrum Communication*. Addison Wesley Longman Publishing Co., Inc., Redwood City, CA, USA.

[311] Viterbo, E. and Boutros, J. (1999). A universal lattice code decoder for fading channels. *Information Theory, IEEE Transactions on*, 45(5):1639–1642.

[312] wai Wong, K., ying Tsui, C., Cheng, R. S. K., and ho Mow, W. (2002). A vlsi architecture of a k-best lattice decoding algorithm for mimo channels. In *2002 IEEE International Symposium on Circuits and Systems. Proceedings (Cat. No.02CH37353)*, volume 3, pages III–273–III–276 vol.3.

[313] Wang, C., Au, E. K. S., Murch, R. D., Mow, W. H., Cheng, R. S., and Lau, V. (2007a). On the performance of the MIMO zero-forcing receiver in the presence of channel estimation error. *IEEE Transactions on Wireless Communications*, 6(3):805–810.

[314] Wang, C., Yan, Y., and Fu, X. (2015a). A high-throughput low-complexity radix-2^4 - 2^2 - 2^3 FFT/IFFT processor with parallel and normal input/output order for IEEE 802.11ad systems. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 23(11):2728–2732.

[315] Wang, C.-Y., Kuo, C.-B., and Jou, J.-Y. (2007b). Hybrid wordlength optimization methods of pipelined FFT processors. *Computers, IEEE Transactions on*, 56(8):1105–1118.

[316] Wang, J., Xiong, C., Zhang, K., and Wei, J. (2016). A mixed-decimation MDF architecture for radix-2^k parallel FFT. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 24(1):67–78.

[317] Wang, M. and Li, Z. (2016). A hybrid SDC/SDF architecture for area and power minimization of floating-point FFT computations. In *2016 IEEE International Symposium on Circuits and Systems (ISCAS)*, pages 2170–2173.

[318] Wang, R. and Giannakis, G. (2006). Approaching MIMO channel capacity with soft detection based on hard sphere decoding. *Communications, IEEE Transactions on*, 54(4):587–590.

[319] Wang, R. and Giannakis, G. B. (2004). Approaching MIMO channel capacity with reduced-complexity soft sphere decoding. In *2004 IEEE Wireless Communications and Networking Conference (IEEE Cat. No.04TH8733)*, volume 3, pages 1620–1625 Vol.3.

[320] Wang, Y., Ge, J. H., Ai, B., Liu, P., and Yang, S. (2004). A soft decision decoding scheme for wireless COFDM with application to DVB-T. *IEEE Transactions on Consumer Electronics*, 50(1):84–88.

[321] Wang, Z., Liu, X., He, B., and Yu, F. (2015b). A combined sdc-sdf architecture for normal I/O pipelined radix-2 FFT. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 23(5):973–977.

[322] Wanhammar, L. (1999). *DSP Integrated Circuits*. Academic Press.

[323] Waters, D. W. and Barry, J. R. (2005). Noise-predictive decision-feedback detection for multiple-input multiple-output channels. *IEEE Transactions on Signal Processing*, 53(5):1852–1859.

[324] Weinstein, S. and Ebert, P. (1971). Data transmission by frequency-division multiplexing using the discrete fourier transform. *IEEE Transactions on Communication Technology*, 19(5):628–634.

[325] Welch, P. D. (1969). A fixed-point fast Fourier transform error analysis. *Audio and Electroacoustics, IEEE Transactions on*, 17(2):151–157.

[326] Wenk, M., Bruderer, L., Burg, A., and Studer, C. (2010). Area- and throughput-optimized VLSI architecture of sphere decoding. In *VLSI System on Chip Conference (VLSI-SoC), 2010 18th IEEE/IFIP*, pages 189–194.

[327] Wenk, M., Zellweger, M., Burg, A., Felber, N., and Fichtner, W. (2006). K-best mimo detection vlsi architectures achieving up to 424 mbps. In *2006 IEEE International Symposium on Circuits and Systems*, pages 4 pp.–1154.

[328] Widrow, B. and Kollár, I. (2008). *Quantization Noise: Roundoff Error in Digital Computation, Signal Processing, Control, and Communications*. Cambridge University Press.

[329] Wiesel, A., Eldar, Y. C., and Shamai, S. (2005). Semidefinite relaxation for detection of 16-QAM signaling in MIMO channels. *IEEE Signal Processing Letters*, 12(9):653–656.

[330] Wiesel, A., Mestre, X., Pages, A., and Fonollosa, J. R. (2003). Efficient implementation of sphere demodulation. In *2003 4th IEEE Workshop on Signal Processing Advances in Wireless Communications - SPAWC 2003 (IEEE Cat. No.03EX689)*, pages 36–40.

[331] Williston, K. (2009). *Digital Signal Processing: World Class Designs*. Elsevier.

[332] Windpassinger, C. and Fischer, R. F. H. (2003). Low-complexity near-maximum-likelihood detection and precoding for MIMO systems using lattice reduction. In *Proceedings 2003 IEEE Information Theory Workshop (Cat. No.03EX674)*, pages 345–348.

[333] Witte, E. M., Borlenghi, F., Ascheid, G., Leupers, R., and Meyr, H. (2010). A scalable vlsi architecture for soft-input soft-output single tree-search sphere decoding. *IEEE Transactions on Circuits and Systems II: Express Briefs*, 57(9):706–710.

[334] Wold, E. H. and Despain, A. M. (1984). Pipeline and parallel-pipeline FFT processors for VLSI implementations. *IEEE Transactions on Computers*, C-33(5):414–426.

[335] Wu, G.-D. and Liu, Y. M. (2009). Radix-22 based low power reconfigurable fft processor. In *2009 IEEE International Symposium on Industrial Electronics*, pages 1134–1138.

[336] Wu, X. and Thompson, J. S. (2011). A fixed-complexity soft-mimo detector via parallel candidate adding scheme and its fpga implementation. *IEEE Communications Letters*, 15(2):241–243.

[337] Wu, Y., Woerner, B. D., and Ebel, W. J. (2000). A simple stopping criterion for turbo decoding. *IEEE Communications Letters*, 4(8):258–260.

[338] Wu (EURASIP Member), D., Eilert, J., Asghar, R., and Liu, D. (2010). VLSI implementation of a fixed-complexity soft-output MIMO detector for high-speed wireless. *EURASIP Journal on Wireless Communications and Networking*, 2010(1):893184.

[339] Wubben, D., Bohnke, R., Kuhn, V., and Kammeyer, K. D. (2003). Mmse extension of v-blast based on sorted qr decomposition. In *2003 IEEE 58th Vehicular Technology Conference. VTC 2003-Fall (IEEE Cat. No.03CH37484)*, volume 1, pages 508–512 Vol.1.

[340] Wubben, D., Bohnke, R., Kuhn, V., and Kammeyer, K. D. (2004a). MMSE-based lattice-reduction for near-ML detection of MIMO systems. In *ITG Workshop on Smart Antennas (IEEE Cat. No.04EX802)*, pages 106–113.

[341] Wubben, D., Bohnke, R., Kuhn, V., and Kammeyer, K. D. (2004b). Near-maximum-likelihood detection of MIMO systems using MMSE-based lattice reduction. In *2004 IEEE International Conference on Communications (IEEE Cat. No.04CH37577)*, volume 2, pages 798–802 Vol.2.

[342] Wubben, D., Bohnke, R., Rinas, J., Kuhn, V., and Kammeyer, K. D. (2001). Efficient algorithm for decoding layered space-time codes. *Electronics Letters*, 37(22):1348–1350.

[343] Wubben, D. and Seethaler, D. (2007). On the performance of lattice reduction schemes for mimo data detection. In *2007 Conference Record of the Forty-First Asilomar Conference on Signals, Systems and Computers*, pages 1534–1538.

[344] Xia, K., Wu, B., Zhou, X., and Xiong, T. (2016). A generalized conflict-free address scheme for arbitrary 2k-point memory-based fft processors. In *2016 IEEE International Symposium on Circuits and Systems (ISCAS)*, pages 2126–2129.

[345] Xiao, X., Oruklu, E., and Saniie, J. (2008). An efficient FFT engine with reduced addressing logic. *IEEE Transactions on Circuits and Systems II: Express Briefs*, 55(11):1149–1153.

[346] Xiao, X., Oruklu, E., and Saniie, J. (2009). Fast memory addressing scheme for radix-4 fft implementation. In *2009 IEEE International Conference on Electro/Information Technology*, pages 437–440.

[347] Yang, B., Letaief, K. B., Cheng, R. S., and Cao, Z. (2001). Channel estimation for OFDM transmission in multipath fading channels based on parametric channel modeling. *IEEE Transactions on Communications*, 49(3):467–479.

[348] Yang, C. H., Yu, T. H., and Markovic, D. (2012). Power and area minimization of reconfigurable FFT processors: A 3GPP-LTE example. *IEEE Journal of Solid-State Circuits*, 47(3):757–768.

[349] Yang, G. and Jung, Y. (2010). Scalable fft processor for mimo-ofdm based sdr systems. In *Wireless Pervasive Computing (ISWPC), 2010 5th IEEE International Symposium on*, pages 517–521.

[350] Yang, H. (2005). A road to future broadband wireless access: MIMO-OFDM-based air interface. *IEEE Communications Magazine*, 43(1):53–60.

[351] Yang, K. J., Tsai, S. H., Chang, R. C., Chen, Y. C., and Chuang, G. C. H. (2013a). VLSI implementation of a low complexity 4×4 MIMO sphere decoder with table enumeration. In *2013 IEEE International Symposium on Circuits and Systems (ISCAS2013)*, pages 2167–2170.

[352] Yang, K. J., Tsai, S. H., and Chuang, G. C. H. (2013b). MDC FFT/IFFT processor with variable length for MIMO-OFDM systems. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 21(4):720–731.

[353] Yang, S. W. and Lee, J. Y. (2014). Constant twiddle factor multiplier sharing in multi-path delay feedback parallel pipelined FFT processors. *Electronics Letters*, 50(15):1050–1052.

[354] Yao, H. and Wornell, G. W. (2002). Lattice-reduction-aided detectors for MIMO communication systems. In *Global Telecommunications Conference, 2002. GLOBECOM '02. IEEE*, volume 1, pages 424–428 vol.1.

[355] Yee, M. S. (2005). Max-log-MAP sphere decoder. In *Proceedings. (ICASSP '05). IEEE International Conference on Acoustics, Speech, and Signal Processing, 2005.*, volume 3, pages iii/1013–iii/1016 Vol. 3.

[356] Yeh, W.-C. and Jen, C.-W. (2003). High-speed and low-power split-radix FFT. *IEEE Transactions on Signal Processing*, 51(3):864–874.

[357] Yi, X., Shieh, W., and Tang, Y. (2007). Phase estimation for coherent optical OFDM. *IEEE Photonics Technology Letters*, 19(12):919–921.

[358] Ylioinas, J. and Juntti, M. (2009). Iterative joint detection, decoding, and channel estimation in turbo-coded MIMO-OFDM. *IEEE Transactions on Vehicular Technology*, 58(4):1784–1796.

[359] Yoshizawa, S., Orikasa, A., and Miyanaga, Y. (2011). An area and power efficient pipeline FFT processor for 8 x 8 MIMO-OFDM systems. In *2011 IEEE International Symposium of Circuits and Systems (ISCAS)*, pages 2705–2708.

[360] Yu, Y.-H., Liu, H.-Y., Hsu, T.-Y., and Lee, C.-Y. (2004). A joint scheme of decision-directed channel estimation and weighted-average phase error tracking for OFDM WLAN systems. In *The 2004 IEEE Asia-Pacific Conference on Circuits and Systems, 2004. Proceedings.*, volume 2, pages 985–988 vol.2.

[361] Yuan, F. L., Lin, Y. H., Wu, C. F., Shiue, M. T., and Wang, C. K. (2008). A 256-point dataflow scheduling 2 x 2 mimo fft/ifft processor for ieee 802.16 wman. In *Solid-State Circuits Conference, 2008. A-SSCC '08. IEEE Asian*, pages 309–312.

[362] Zhang, H. and Xia, X.-G. (2006). Iterative decoding and demodulation for single-antenna vector OFDM systems. *IEEE Transactions on Vehicular Technology*, 55(4):1447–1454.

[363] Zhang, J., Chen, S., Mu, X., and Hanzo, L. (2014). Evolutionary-algorithm-assisted joint channel estimation and turbo multiuser detection/decoding for OFDM/SDMA. *IEEE Transactions on Vehicular Technology*, 63(3):1204–1222.

[364] Zhang, Y. J. and Letaief, K. B. (2005). An efficient resource-allocation scheme for spatial multiuser access in MIMO/OFDM systems. *IEEE Transactions on Communications*, 53(1):107–116.

[365] Zonst, A. E. (2004). *Understanding FFT Applications - A Tutonal for Laymen, Students, Technicians and Working Engineers*. Citrus Press.

[366] Zubow, A., Marotzke, J., Camps-Mur, D., and Perez-Costa, X. (2012). sGSA: An SDMA-OFDMA scheduling solution. In *European Wireless 2012; 18th European Wireless Conference 2012*, pages 1–8.