Advanced Multiple Access Techniques for Ultra-Reliable Low-Latency Communications

DU, Yuyang

A Thesis Submitted in Partial Fulfilment
of the Requirements for the Degree of
Doctor of Philosophy
in
Information Engineering

The Chinese University of Hong Kong
July 2025

Abstract of thesis entitled:
Advanced Multiple Access Techniques for Ultra-Reliable Low-Latency Communications
Submitted by DU, Yuyang
for the degree of Doctor of Philosophy
at The Chinese University of Hong Kong in July 2025

Ultra-Reliable Low-Latency Communication (URLLC) is a critical requirement in next-generation wireless networks. URLLC plays a vital role in enabling applications such as industrial automation, vehicular communications, and massive Internet of Things (IoT). This thesis investigates advanced multiple access techniques to improve the reliability, efficiency, and deployability of URLLC systems, focusing on two key schemes: Interleaved Frequency Division Multiple Access (IFDMA) and Carrier Sense Multiple Access (CSMA). Additionally, this thesis explores the use of AI-driven methodologies to accelerate the deployment of URLLC systems on FPGA-based Software-Defined Radios (SDRs).

First, this thesis addresses the computational challenges in IFDMA systems. This thesis demonstrates that the complexity of Fast Fourier Transform (FFT) computation in IFDMA can be significantly reduced through a pruning algorithm tailored to the “compact” structure of IFDMA transceivers. Building upon the pruned FFT model, a novel scheduling strategy, referred to as the Multi-Priority Scheduling (MPS) scheme, is proposed. MPS analyzes the precedence constraints within the pruned FFT computation graph and optimizes the execution of butterfly operations to improve processor utilization. The proposed pruning and scheduling techniques substantially reduce computation latency, making IFDMA more suitable for low-latency applications.

Second, this thesis tackles the problem of packet detection in CSMA, a largely overlooked reliability issue despite the widespread adoption of CSMA in machine-type communications. Traditional packet detection algorithms, such as the Schmidl & Cox (S&C) method, suffer from analytical intractability. To address this, this thesis introduces a novel compensated autocorrelation metric to enable closed-form expressions for both false-alarm and missed-detection probabilities, thereby allowing rigorous performance evaluation. A Pareto-based benchmarking methodology is also proposed to fairly compare detection algorithms by jointly considering both error types. Furthermore, this thesis puts forth a real-part-only variant of the S&C algorithm (RP-S&C), which significantly enhances a CSMA system's detection performance under practical conditions by exploiting modern hardware's improved oscillator stability. This packet detection scheme is further extended to multi-antenna systems and extensively evaluated in realistic distributed antenna scenarios. These contributions greatly improve the robustness of packet detection in CSMA-based URLLC systems.

Finally, this thesis explores the potential of Large Language Models (LLMs) to accelerate the development of FPGA-based URLLC systems. Implementing advanced multiple access techniques in hardware requires low-level programming in hardware description languages (HDLs) such as Verilog, which is time-consuming and error-prone. To overcome this challenge, this thesis introduces two novel prompting strategies, i.e., Chain-of-Thought (CoT) reasoning and Iterative In-Context Learning (IICL), to enhance LLMs' ability to generate Verilog code for signal processing modules. Case studies on FFTs demonstrate that these prompting techniques can significantly improve LLM's ability in generating Verilog codes, enabling more efficient deployment of optimized multiple access systems.

In general, this thesis presents a comprehensive study for advancing multiple access in URLLC by optimizing IFDMA and CSMA protocols and leveraging AI-driven automation for hardware design. The proposed solutions contribute to the development of low-latency, high-reliability wireless communication systems and facilitate the rapid realization of next-generation network technologies.

摘要

超可靠低时延（Ultra-Reliable Low-Latency Communication，URLLC）是下一代无线网络的核心需求，在工业自动化、车联网通信和大规模物联网（IoT）等关键应用中发挥着至关重要的作用。本文围绕两种关键的多址接入技术–交错频分多址（Interleaved Frequency Division Multiple Access，IFDMA）和载波侦听多址（Carrier Sense Multiple Access，CSMA）展开研究，旨在提升 URLLC 系统的可靠性、效率与可部署性。此外，本文还探索了基于人工智能的方法在 FPGA 平台上加速 URLLC 系统部署的可能性，特别是在基于软件定义无线电（SDR）的架构中。

首先，本文解决了 IFDMA 系统中的计算复杂度问题。针对 IFDMA 收发器结构的“紧凑性”，设计了一种剪枝算法以降低快速傅里叶变换（Fast Fourier Transform，FFT）的计算复杂度。在此基础上，本文提出一种名为多优先级调度（Multi-Priority Scheduling，MPS）的新型调度策略，分析剪枝后的 FFT 计算图中的优先关系，优化蝶形运算的执行顺序，从而提升处理器资源的利用率。所提出的剪枝与调度方法显著降低了 IFDMA 的计算时延，使其更适用于低时延场景。

其次，本文针对 CSMA 系统中被长期忽视的封包检测问题展开研究。尽管 CSMA 在机器类通信中被广泛采用，但传统的封包检测算法（如 Schmidl & Cox 方法）在理论分析上难以处理。为此，本文提出一种新的“补偿自相关”检测指标，使得误警率与漏检率可以通过闭式表达式精确计算，从而建立起严格的理论分析框架。同时，本文引入基于帕累托最优性的基准评估方法，可公平比较不同检测算法在误警与漏检之间的权衡效果。进一步地，本文提出实部自相关（RP-S&C）算法，以利用现代硬件振荡器稳定性提升带来的优势，有效降低误警率并提升检测性能。该检测方案还被推广至多天线系统，并在真实分

布式天线场景下进行了系统评估，显著提升了 CSMA 系统中封包检测的鲁棒性与可靠性。

最后，本文探索了利用大型语言模型（Large Language Models, LLMs）加速 FPGA 平台上 URLLC 系统开发的潜力。由于硬件实现过程中需要使用如 Verilog 等硬件描述语言（HDL）进行底层编程，开发过程通常十分耗时且容易出错。为此，本文提出两种创新的提示策略：链式思维（Chain-of-Thought, CoT）推理与迭代式上下文学习（Iterative In-Context Learning, IICL），以增强 LLM 在自动生成 FPGA 信号处理模块代码方面的能力。以 FFT 模块为例的实验证明，这些 AI 提示技术能显著提升 LLM 生成 Verilog 代码的质量和效率，促进多址接入系统的快速部署。

综上所述，本文从 IFDMA 和 CSMA 两种接入方式出发，结合人工智能辅助的硬件自动化设计，提出了一套完整的 URLLC 多址接入优化框架。所提出的方法为构建低时延、高可靠的无线通信系统提供了理论支撑与实践路径，推动了下一代网络技术的高效实现。

Publications by Topic

Ultra-Reliable Low-Latency Communication

- **Y. DU** and S. C. Liew, Reliable Packet Detection in Random Access Networks: Analysis, Benchmark, and Optimization, in *IEEE Transactions on Vehicular Technology*, 2025.
- **Y. Du**, S. C. Liew. and Y. Shao, Efficient FFT computation in IFDMA transceivers, in *IEEE Transactions on Wireless Communications*, 2024.
- **Y. Du**, H. Deng, S. C. Liew, K. Chen, Y. Shao and H. Chen, The power of large language models for wireless communication system development: A case study on FPGA platforms, in *IEEE VTC-Fall*, 2024.
- **Y. Du**, L. Hao and Y. Lei, Nonlinear Multi-Carrier System with Signal Clipping: Measurement, Analysis, and Optimization, in *IEEE Systems Journal*, 2024.
- **Y. Du**, L. Hao and Y. Lei, SER Optimization in OFDM-IM Systems with Nonlinear Power Amplifiers, in *IEEE Transactions on Vehicular Technology*, 2023.
- **Y. Du**, L. Hao and Y. Lei, SER Analysis and Joint Optimization in Nonlinear MIMO-OFDM Systems with Clipping, in *IEEE VTC-Spring*, 2023.
- **Y. Du**, Y. Lei and S. McGrath, SER optimization in transparent OFDM relay systems in the presence of dual nonlinearity, in

Digital Signal Processing, 2022.

- **Y. Du**, J. Chen, Y. Lei and X. Hao, Performance analysis of nonlinear spatial modulation multiple-input multiple-output systems, in *Digital Signal Processing*, 2021.

- **Y. Du**, L. Hao, Z. Liu, Y. Chen and Y. Lei, Ergodic Rate Performance in Nonlinear Omnidirectional Coding MIMO-OFDM Systems, in *IEEE UEMCON*, 2019.

AI for Networking & Networking for AI

- L. Wang*, X. Long*, **Y. Du***, X. Liu, K. Chen and S. C. Liew, Cellular-X: An LLM-empowered Cellular Agent for Efficient Base Station Operations, in *ACM MobiSys*, 2025.

- F. Zhang*, **Y. Du***, K. Chen*, Y. Shao and S. C. Liew, Out-of-Distribution in Image Semantic Communication: A Solution with Multimodal Large Language Models, in *IEEE Transactions on Machine Learning in Communications and Networking*, 2025

- L. Wang*, **Y. Du***, J. Lin*, K. Chen and S. C. Liew, Rephrase and Contrast: Fine-Tuning Language Models for Enhanced Understanding of Communication and Computer Networks, in *IEEE ICNC*, 2025.

- H. Cui*, **Y. Du***, Q. Yang*, Y. Shao and S. C. Liew, LLMind: Orchestrating AI and IoT with LLMs for complex task execution, in *IEEE Communications Magazine*, 2024.

- F. Zhang*, **Y. Du***, K. Chen*, Y. Shao and S. C. Liew, Addressing Out-of-Distribution Challenges in Image Semantic Communication Systems with Multi-modal Large Language Models, in *IEEE WiOpt*, 2024.

Others

- **Y. Du**, K. Chen, Y. Zhan, C. H. Low, M. Islam, Z. Guo, Y. Jin, G. Chen and P. A. Heng, LMT++: Adaptively Collaborating LLMs with Multi-specialized Teachers for Continual VQA in Robotic Surgical Videos, in *IEEE Transactions on Medical Imaging*, 2025.
- Y. Zhang, X. Chen, K. Chen#, **Y. Du#**, X. Dang, P. A. Heng The Dual-use Dilemma in LLMs: Do Empowering Ethical Capacities Make a Degraded Utility, https://arxiv.org/abs/2501.13952, 2025.
- K. Chen*, **Y. Du***, T. You, M. Islam, Z. Guo, Y. Jin, G. Chen and P. A. Heng, LLM-Assisted Multi-Teacher Continual Learning for Visual Question Answering in Robotic Surgery, in *IEEE ICRA*, 2024.
- K. Chen, H. Cao, J. Li, **Y. Du**, M. Guo, X. Zeng, L. Li, J. Qiu, P. A. Heng, and G. Chen An Autonomous Large Language Model Agent for Chemical Literature Data Mining, https://arxiv.org/abs/2402.12993, 2024.
- K. Chen, J. Li, K. Wang, **Y. Du**, J. Yu, J. Lu, L. Li, J. Qiu, P. A. Heng, G. Chen Chemist-X: Large Language Model-empowered Agent for Reaction Condition Recommendation in Chemical Synthesis, http://arxiv.org/abs/2311.10776, 2023.
- Y. Shao, S. C. Liew, H. Chen and **Y. Du**, Flow sampling: Network monitoring in large-scale software-defined IoT networks, in *IEEE Transactions on Communications*, 2021.

Acknowledgement

With deep appreciation, I would like to acknowledge my supervisor, Prof. Soung Chang Liew, for his invaluable guidance throughout my PhD journey. I still vividly remember how, in July 2020, when I had the honor of receiving an offer from him, Prof. Liew described our research group: curiosity, keenness for problem solving, and personal growth. As my thesis advisor, Prof. Liew has exemplified these ideals throughout my PhD training. Prof. Liew has been an ideal mentor in every regard - providing me with hands-on technical training and thoughtful professional advice. His timely support and insightful suggestions have been instrumental in shaping the direction and quality of my research. Under his leadership, our group has maintained an open, free, and harmonious atmosphere, which made my PhD experience both warm and memorable.

My sincere thanks are also reserved for Prof. He Chen, who has devoted his time to helping me explore new directions at the early stage of my research journey. I have enjoyed numerous discussions with him on various kinds of issues. I would also like to acknowledge Prof. Yiming Lei from Peking University and Prof. Minlan Yu from Harvard University for their vital roles in my academic journey.

Heartfelt thanks should be conveyed to members of the Advanced Wireless System Group (AWSG). I would like to thanks to Dr. Yulin Shao, Dr. Jiaxin Liang, Dr. Gongpu Chen, Dr. Lihao Zhang, Dr. Taotao Wang, Dr. Lizhao You, Dr. Haiyang Xin. As alumni of AWSG, they have been providing me with valuable suggestions and immeasurable help throughout my PhD effort. Also, I am so grateful to Qun

Yang, Hongwei Cui, Liujianfu Wang, Shiqi Xu, Feifan Zhang, my colleagues at AWSG. I enjoy all the time working with them to explore unknown.

I thank my parents for their care and support throughout my entire life. They gave me everything they have and treat me so well. I am also thankful to my wife and my closest research collaborator, Kexin Chen, for filling my life with love, hope, and wisdom. Without her understanding and support, it would not have been possible to finish this Ph.D. study.

To those who walked with me on this journey.

Contents

Abstract i

Publications vi

Acknowledgement ix

1 Introduction 1

- 1.1 Multiple Access for URLLC 2
- 1.2 Interleaved Frequency Division Multiple Access 4
- 1.3 Carrier Sense Multiple Access 7
- 1.4 FPGA-Based Software-Defined Radio Systems for Fast Prototyping 9
- 1.5 Thesis Organization 10

2 Efficient FFT Computation in IFDMA 11

- 2.1 Background 11
  - 2.1.1 Quick Review and Motivation 13
  - 2.1.2 Related Works 16
  - 2.1.3 Contributions and Findings 17
- 2.2 Definitions and Problem Formulation 19
  - 2.2.1 Problem Formulation 19
  - 2.2.2 Feasibility of a Task-execution Schedule 23
  - 2.2.3 Optimality of a Heuristic Scheduling Algorithm 24
- 2.3 Multi-Priority Scheduling for IFDMA FFT 28
  - 2.3.1 Structural Properties of FFT Precedence Graphs 29

    - 2.3.2 The Heuristic Scheduling Algorithm 30
  - 2.4 Lower Bound Analysis for IFDMA FFT Computing . . 35
  - 2.5 Experimental Results 39
    - 2.5.1 Optimality of MPS 40
    - 2.5.2 Applying MPS-FFT in Compact IFDMA Transceivers 41
  - 2.6 Summary 45

3 Reliable Packet Detection in CSMA **47**
  - 3.1 Background 47
  - 3.2 RP-S&C for Single-antenna Packet Detection 52
    - 3.2.1 Conventional S&C Algorithm 52
    - 3.2.2 Our Improvement over the Conventional S&C Algorithm 54
    - 3.2.3 Analysis of RP-S&C with Gaussian Approximation 57
    - 3.2.4 Validating Gaussian Assumption 62
  - 3.3 Benchmarking RP-S&C With the Conventional S&C Algorithms 64
  - 3.4 RP-S&C for Multi-Antenna Packet Detection: Analysis and Optimization 69
    - 3.4.1 Problem Formulation 69
    - 3.4.2 Optimal Weights for False Alarm (WFA) and Missed Detection (WMD) 71
  - 3.5 RP-S&C for Multi-Antenna Packet Detection: Experiments and Discussions 76
    - 3.5.1 Implementation Issues 77
    - 3.5.2 Benchmark WFA and WMD in two typical DAS scenarios 78
    - 3.5.3 Benchmark Results in a Distributed Antenna Dataset 82
  - 3.6 Summary 87

4 LLM for FPGA-based SDR **89**
  - 4.1 Background 89

4.2 Related Works 91
4.3 Existing Challenges 92
4.3.1 Challenge One: Multi-step Reasoning 93
4.3.2 Challenge Two: Subtask Execution Scheduling . 94
4.4 Chain-of-Thought for Multi-Step Reasoning 98
4.5 Iterative In-context Learning for Task Schedule 101
4.6 Summary 105

5 Conclusion 108
5.1 Conclusion Remarks 108
5.2 Future Works 109

A Isomorphism of FFT Precedence Graphs 111

B Recursion Function of f_n 115

C Test Set Generation Policy for $n \geq 7$ 118

D Pseudo Code for MPS-FFT 121

E The Computation of IFDMA FFT is Computation Bounded123

F CFO's influence on autocorrelation 125

G The Optimal packet-detection threshold 127

H Packet Detections in Slow Fading Environments 130

I Pareto Comparison versus Classification 132

Bibliography 134

List of Figures

1.1 A background experiment conducted in [1], which compared the PAPR of wireless signals transmitted by an IFDMA/LFDMA/OFDMA system. Here the performance is measured by the complementary cumulative distribution function (CCDF) of PAPR. CCDF is defined as the probability that the PAPR is higher than a certain threshold η_0 in dB, i.e., Pr(PAPR>η_0). From the image, we see that IFDMA has significantly lower PAPR than the two counterparts. Readers are referred to Section V-B for [1] for detailed configuration of the experiment presented. 6

1.2 The structure of a conventional IFDMA transceiver with only a single stream, in which a N-point DFT plus a M-point IDFT are needed at the transmitter side, and a M-point DFT plus a N-point IDFT are needed at the receiver side [1]. If multiple users are considered, IDFTs of different sizes are needed at the transmitter side, and DFTs of different sizes are needed at the receiver side. . 6

2.1 An example on how the receiver extracts/demultiplexes the IFDMA data streams embedded in IFDMA signals. 13

2.2 An example on a compact IFDMA FFT network with two users. 14

2.3 An example to illustrate how to obtain an FFT precedence graph given $\{S_0, S_1, ...S_{R-1}\}$. 20

2.4 Legitimate parent-child structures within the FFT precedence graph. 30
2.5 Evolution of the FFT precedence graph during the execution of the heuristic algorithm, assuming three processors. The vertexes enclosed by a solid boundary belong to the precedence graph. The dashed vertexes do not belong to the precedence graph, and they are shown here for illustrative purposes only. 31
2.6 An example of "trunk vertexes" and "branch vertexes". 36
2.7 Computation complexity of compact IFDMA transceiver v1 and compact IFDMA transceiver v2. 43
2.8 Experimental results showing that MPS-FFT outperforms pipelined FFT in terms of hardware utilization. . 45

3.1 A general packet format for random access. 52
3.2 Illustration of how S&C detects wireless packets with an example of missed packet detection. Here $\eta = 16$. . . . 53
3.3 Illustration of how false alarms arise. 53
3.4 (a) Simulated and analytical results of false-alarm probability under various η and ρ settings. (b) Simulated and analytical results of missed-detection probability under various η, γ, and ρ settings. 63
3.5 Two example MD-FA curves for illustrating our "Pareto comparison" benchmarking method. 65
3.6 (a) An example to illustrate that one scheme is superior to the other within a specific region of interest. (b) An example to illustrate that one scheme is comparable with the other within a specific region of interest. The green dashed lines in the two figures demarcate the region of interest. 67
3.7 RP-S&C versus conventional S&C under various CFO and SNR conditions. 68

3.8 (a) MD-FA curves of WFA/I-WMD/P-WMD in the non-blocked scenario. (b) MD-FA curves of WFA/I-WMD/P-WMD in the partially blocked scenario. 81
3.9 The location of Group A/B/C/D and an example of a moving transmitter. 83
3.10 A visual illustration of SNR correlations between different antennas (using $Ant(A, 1)$ as a reference). 84
3.11 (a) The scatter plot of SNRs in $Ant(A, 1)$ and $Ant(A, 2)$. (b) The scatter plot of SNRs in $Ant(A, 1)$ and $Ant(D, 6)$. In (b), the "fully blocked zone" refers to the SNR region where both antennas have very low SNR. Both two figures use the original SNR values rather than the dB values. 85

4.1 A failed prompt for generating the FFT module. 93
4.2 A typical GPT4-written Verilog implementation for eight-point FFT. 95
4.3 Error rates of different LLMs in twiddle factor generations. 96
4.4 Precedence graphs of an eight-point FFT. The output of the green and the blue butterfly computations (in the first FFT stage) serve as the input for the red butterfly computation (in the second FFT stage). We cannot execute the red task until the blue and green tasks have been completed, indicating a precedence relationship between the red task and the blue/green task. We refer readers to Chapter 2.2.3 for detailed explanation of the parallel/precedence relationship. 97
4.5 Error rates of different LLMs in scheduling butterfly computations within the FFT tasks. 98
4.6 A example to show how CoT enhance LLMs' performance in multi-step mathematical reasoning. 99

4.7 The CoT prompt we used to generate the 32-bit sequences for twiddle factors in a 16-point FFT. 99
4.8 GPT-4 model's performance with the CoT prompting scheme. 100
4.9 Illustrations of conventional ICL prompting scheme. . . 102
4.10 The iterative flow graph of an example eight-point FFT. 103
4.11 Implementation suggestion given to the LLM, which is related to the connection of "enable" and "output valid" signals. Here, we are telling the LLM how to build a four-point FFT by properly connecting the two two-point FFT modules provided. 104
4.12 An illustration of IICL prompting in our FFT generation task. 106

A.1 FFT precedence graphs and corresponding splitting trees for bin allocation $(2, 2, 1, 1, 1, 1)$, $(1, 1, 1, 1, 2, 2)$, $(2, 1, 1, 2, 1, 1)$, $(2, 1, 1, 1, 1, 2)$, $(1, 1, 2, 1, 1, 2)$, and $(1, 1, 2, 2, 1, 1)$. . . . 112

B.1 An illustration of how non-single-stream non-isomorphic instances for eight-subcarrier IFDMA is generated. . . 116

G.1 How the LP problem could be solved. 128

List of Tables

2.1 Number of non-isomorphic instances for various n . . . 26
2.2 η_n and γ_n for the scheduling problems in a 2^n-point IFDMA FFT (the case of an arbitrary number of processors) 41
2.3 η_n and γ_n for the scheduling problems in a 2^n-point IFDMA FFT (the case of a power-of-two number of processors) 42
2.4 Complexities of Conventional IFDMA Transceivers and Compact IFDMA Transceiver v1 43
2.5 Number of Average Time Slots Needed for the FFT computation in Compact IFDMA Transceivers v1/v2 . . . 46

3.1 SNR conditions of two typical DAS examples. 80
3.2 Benchmarking WFA and I-WMD/P-WMD with DICHASUS dataset. 86

B.1 Non-isomorphic instances for the fully packaged compact IFDMA system with four subcarriers 115
B.2 Non-isomorphic instances for the fully packaged compact IFDMA system with eight bins 117
B.3 Number of non-isomorphic instances for $n = 1, 2, ..., 10$ 117

Chapter 1

Introduction

Ultra-Reliable and Low-Latency Communication (URLLC) is a critical communication paradigm envisioned for future wireless systems, underpinning mission-critical applications such as industrial automation, autonomous driving, and remote medical operations. These applications demand an unprecedented level of reliability and responsiveness to ensure safe and efficient operation. While significant progress has been made in achieving these goals within the framework of 5G/6G standards, the exploration of alternative approaches tailored to diverse application scenarios remains an important and open research direction.

This thesis focuses on advancing multiple access mechanisms to address the unique challenges of URLLC in future wireless systems, which are not confined to the traditional multiple access schemes defined in the 3GPP 5G or future 6G standards but instead explore alternative approaches tailored to diverse application scenarios and requirements. By investigating the latency and reliability issues in Interleaved Frequency Division Multiple Access (IFDMA) and Carrier Sense Multiple Access (CSMA), as well as exploring the efficient implementation of advanced multiple access techniques on hardware platforms, this work explores the potential of leveraging alternative multiple access schemes beyond the existing 3GPP standards to support future URLLC systems.

1.1 Multiple Access for URLLC

A fundamental aspect of enabling URLLC is the efficient coordination of how multiple devices access the wireless medium, commonly referred to as multiple access. In wireless networks, multiple access schemes determine how different users share time, frequency, and spatial resources to transmit and receive data.

In non-URLLC applications, multiple access schemes are typically optimized for higher spatial efficiency and throughput. URLLC, on the other hand, introduces stringent requirements for reliability and latency, necessitating multiple access solutions that go beyond traditional designs focusing on efficiency only. For example, access delays caused by scheduling procedures, contention, or collisions can significantly impact URLLC performance. As such, multiple access becomes a critical enabler of URLLC, directly influencing how quickly and reliably a device can communicate with another.

This thesis investigates three interconnected areas in the context of multiple access for enabling URLLC, addressing both theoretical and practical challenges. Each area targets a critical aspect of URLLC systems, and together, they form a comprehensive framework for advancing multiple access techniques in URLLC.

To begin with, the thesis studies the latency issue in IFDMA, a promising multiple access technique known for its low peak-to-average power ratio (PAPR). This thesis reduces the computation latency in IFDMA with a novel pruning algorithm (for reduced computational workload) and a novel scheduling scheme (for enhanced computational efficiency). These enhancements help IFDMA to better address the stringent delay requirement of URLLC when deployed in future mission-critical communication system.

Meanwhile, the thesis investigates the reliability issue in CSMA. CSMA is a widely used multiple access scheme in machine-type communications, especially for scenarios requiring massive connectivity. Unlike Time Division Multiple Access (TDMA), where resource allocation is deterministic and collision-free, CSMA employs a contention-based approach where devices sense the channel before transmitting. While this decentralized nature makes CSMA highly scalable and efficient for many applications, it also introduces challenges in ensuring reliable communication under URLLC constraints. One major issue is packet detection, a critical step before packet decoding. This thesis notices the reliability issue in CSMA and introduces 1) a novel compensated autocorrelation metric, and 2) a real-part-only variant of the conventional Schmidl & Cox (S&C) algorithm, referred to as the real-part S&C (RP-S&C) algorithm, to improve packet detection accuracy in CSMA systems. These research efforts are crucial for ensuring reliable communication in random access systems under URLLC constraints.

Finally, the thesis explores the efficient realization of advanced multiple access algorithms on field-programmable gate array (FPGA) platforms. While this part of the work does not directly address URLLC-specific challenges, it represents an additional effort to bridge the gap between theoretical advancements and practical deployment by accelerating the implementation of advanced signal processing algorithms within a multiple access scheme on hardware platforms. Nowadays, system prototyping has become an important research method to validate novel algorithms in real-world scenarios. Demonstrating the advanced multiple access techniques developed in this thesis often requires algorithm deployment on high-performance hardware platforms such as FPGA. However, FPGA development is a time-consuming and error-prone process due to the need for low-level hardware description languages (HDLs). To address this, this thesis leverages the capabilities of large language models (LLMs) to automate and accelerate the FPGA development process. By introducing Chain-of-Thought (CoT) reasoning and Iterative In-Context Learning (IICL) prompting strategies, this work demonstrates how LLMs can generate high-quality Verilog code for complex signal processing modules, such as FFTs. These techniques, while not directly tied to URLLC, serve as a complementary contribution by facilitating the rapid realization of the URLLC systems proposed in this thesis.

While the above three contributions may seem distinct in their focuses, they fundamentally transform existing multiple access mechanisms from the latency and the reliability perspectives to improve their performance in URLLC scenarios, ultimately converging on the critical question of how to efficiently implement these algorithms in real-world systems. Together, these contributions form a comprehensive investigation on multiple access mechanism for URLLC, covering key aspects ranging from algorithm designs to system implementations.

The following of the introduction chapter starts with an overview of the two vital multiple access schemes to be investigated in this thesis, and this is followed by a quick introduction of about the validation of real-time URLLC systems on software-defined radio (SDR) platforms built upon FPGA chips.

1.2 Interleaved Frequency Division Multiple Access

IFDMA is a broadband signal modulation and multiple-access technology with significant potential for advanced wireless communication systems. Compared with other widely adopted technologies, such as orthogonal FDMA (OFDMA) [2, 3], a salient feature of IFDMA is its inherently low Peak-to-Average Power Ratio (PAPR) [4], which makes it highly energy-efficient and particularly suitable for power-constrained devices, such as battery-powered IoT sensors and wearable devices. This characteristic is also valuable in the context of URLLC, given that high PAPR may also introduce strong nonlinearity at the RF power amplifiers, significantly influencing the system's signal integrity, which plays a critical role in ensuring reliable communications.

Although IFDMA has not been adopted in current 3GPP standards, its unique advantage in PAPR positions it as a promising candidate for future wireless systems, particularly in scenarios requiring highly reliable performance under stringent energy constraints. By addressing the computational challenges associated with IFDMA implementation and exploring its integration into emerging industrial and IoT applications, the research in this thesis lays the groundwork for unlocking its full potential in next-generation communication networks.

Fig. 1.1 presents the experimental results of a previous study [1], in which the PAPR of IFDMA is compared with 1) the PAPR of OFDMA, a multiple access scheme known for its wide applications in 5G/6G and the latest WiFi standard, and 2) the PAPR of localized FDMA, which is known for its significantly lower PAPR than OFDMA. This background experiment serves as a good illustration of IFDMA's advantage in low PAPR, showing that IFDMA has a significantly reduced PAPR performance compared to the widely used OFDMA and the advanced LFDMA scheme. This low-PAPR advantage brings lower signal distortion and higher power efficiency when the signal is amplified by a power amplifier [5], making IFDMA a promising technology for battery-charged portable communication devices such as smartphones or laptops, in which energy efficiency is a very important concern [6].

Previous investigations of IFDMA focused mainly on the transceiver complexity. In [1, 7], a new class of IFDMA transceivers that is significantly less complex than conventional IFDMA transceivers has been developed. The design in [1, 7] allows a single IFFT/FFT module to perform the multiplexing/demultiplexing of multiple IFDMA data streams of different sizes, while conventional IFDMA transceivers require multiple IFFT/FFT modules for multiplexing/demultiplexing. Fig. 1.2 illustrates the structure of conventional IFDMA transceivers.

This new IFDMA transceiver design, referred to as the "compact IFDMA" to capture the fact that it performs IFDMA functions in a compact and efficient manner, obviates the need to have IFFT/FFT modules of different sizes to cater to different IFDMA data streams. The key is to insert (for multiplexing at the transmitter) and extract (for demultiplexing at the receiver) the IFDMA data streams at different stages of the IFFT/FFT module according to their sizes.

We have noticed that, although [1] and [7] have laid down the theo-

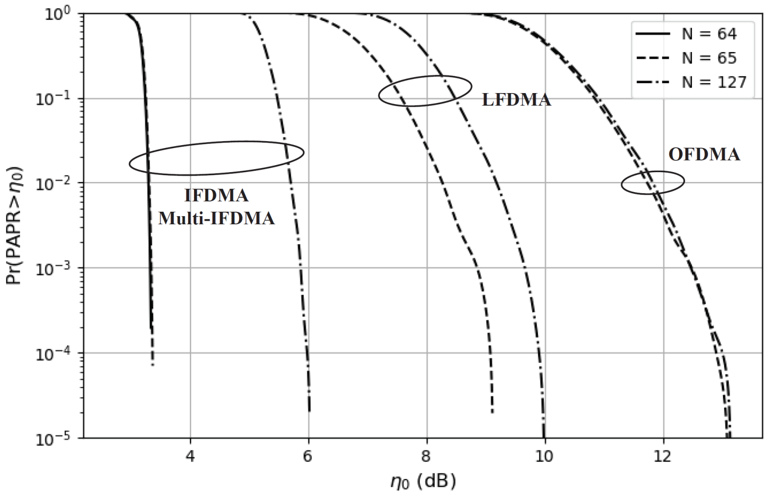


Figure 1.1: A background experiment conducted in [1], which compared the PAPR of wireless signals transmitted by an IFDMA/LFDMA/OFDMA system. Here the performance is measured by the complementary cumulative distribution function (CCDF) of PAPR. CCDF is defined as the probability that the PAPR is higher than a certain threshold η_0 in dB, i.e., Pr(PAPR>η_0). From the image, we see that IFDMA has significantly lower PAPR than the two counterparts. Readers are referred to Section V-B for [1] for detailed configuration of the experiment presented.

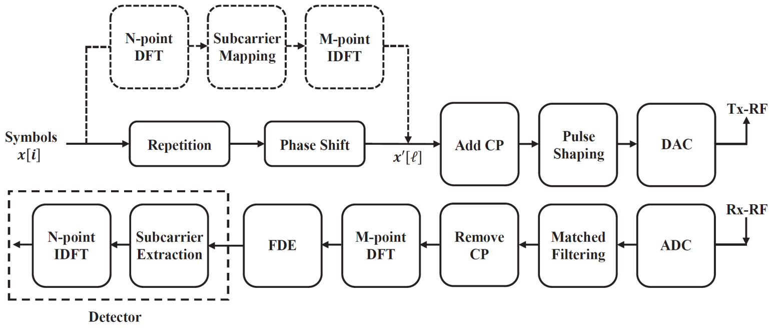


Figure 1.2: The structure of a conventional IFDMA transceiver with only a single stream, in which a N-point DFT plus a M-point IDFT are needed at the transmitter side, and a M-point DFT plus a N-point IDFT are needed at the receiver side [1]. If multiple users are considered, IDFTs of different sizes are needed at the transmitter side, and DFTs of different sizes are needed at the receiver side.

retical foundation for compact IFDMA, its implementation in practical hardware under a resource constraint, such as an application-specific integrated circuit (ASIC) or an FPGA chip, presents several technical challenges yet to be addressed. In [1, 7], the authors assumed perfect parallelism in the compact IFDMA FFT computation (i.e., an unlimited number of processors).

However, when the computation of IFDMA is to be implemented in an ASIC or FPGA, having abundant hardware to process the butterfly computations in parallel may not be possible, particularly when the number of subcarriers is large. That is, saving the resource utilization is a vital issue in ASIC or FPGA design [8], and the practical engineering of compact IFDMA may be forced to use fewer processors for butterfly computations. As a result, the amount of parallelism that can be achieved is limited, making computation latency a major issue for the IFDMA transceiver.

Using the FFT computation as a representative, Chapter 2 shows how the latency issue can be addressed by 1) pruning the FFT structure and 2) scheduling the butterfly computation processes within the FFT module specially pruned for IFDMA. These investigations significantly reduce the computation latency of IFDMA, making this energy-efficient technique more suitable for URLLC applications.

1.3 Carrier Sense Multiple Access

Wireless random access protocols, such as CSMA, offer significant advantages for mission-critical applications [9]. Unlike centralized access control mechanisms, CSMA enables massive machine-type communications (MTC) intrinsic to smart transportation or industrial networks without requiring a predetermined transmission schedule. For instance, in traffic monitoring systems, a sensor embedded within infrastructure or onboard a vehicle may generate a new packet sporadically, such as upon detecting an anomaly, and then transmit this information to a central monitoring system through the wireless channel. In such cases, where transmission is rare and unpredictable, employing CSMA is more efficient than preallocating dedicated wireless resources (e.g., time slots or subcarriers) to each sensor. Moreover, centralized access mechanisms, such as TDMA or Frequency Division Multiple Access (FDMA), become impractical when the number of connected devices exceeds the available wireless resources.

While most research and development efforts for URLLC are centered around 5G technologies, this thesis investigates the feasibility of achieving URLLC in the context of WiFi, which inherently adopts CSMA as its underlying multiple access protocol. WiFi offers several advantages, particularly in industrial environments where its operation in unlicensed spectrum reduces deployment costs and enables flexible and scalable deployments. However, ensuring reliability in WiFi-based URLLC systems requires addressing the inherent challenges of CSMA.

In CSMA, a receiver does not know when a wireless device will transmit a packet to it. To conserve power and avoid being wrongly occupied, the packet decoding circuitry of a receiver should not get activated unless a packet is being transmitted, i.e., the receiver needs to detect the incoming packet before decoding it. Therefore, there are two primary causes for reception failures in CSMA: 1) the payload of the packet cannot be decoded due to noise or transmission errors, or 2) the payload of the packet can be decoded with advanced decoding schemes, but the packet is not even detected.

A recent trend in the URLLC research is to use packets with short payloads to reduce decoding error probability [10]. These research efforts significantly overcome the first type of failure but ignore possible reception failure in the packet detection process. This thesis is an attempt to address the second type of failure and enhance the overall reliability in CSMA networks. This thesis revisits the conventional Schmidl-and-Cox (S&C) algorithm [11] widely applied as the underlying scheme for many recent packet-detection studies [12, 13, 14, 15, 16, 17, 18, 19, 20, 21]. Chapter 3 of this thesis points out the limitations of this conventional algorithm, as well as related prior works on the conventional detection algorithm, and puts forth novel solutions for these limitations.

1.4 FPGA-Based Software-Defined Radio Systems for Fast Prototyping

Systematic validation is an important next step after advanced multiple access schemes are developed. In wireless communication research, implementing a prototype on SDR platforms has always been a convenient way for the system validation. In recent years, FPGAs have emerged as a popular SDR platform due to their significant reduction of processing latency compared to the conventional Universal Software Radio Peripheral (USRP) platform.

While FPGAs provide a versatile platform for system realization, their implementation poses a much steeper learning curve than conventional SDR platforms. Unlike traditional SDRs, for which high-level languages like C or Python are commonly used, FPGAs demand a more direct interaction with hardware through hardware description languages (HDLs) such as Verilog. The low-level programming, as well as the synthesis-friendly programming style required in HDL [22], necessitate a deep understanding of signal processing and internal hardware operations, thus prolonging the development cycle of SDR designs.

Recent advancements in large language models (LLMs) suggest a potential AI-assisted solution to accelerate HDL developments. Prior works have explored the use of LLMs for automated bug repairs [23], basic Verilog coding abilities [24], power-performance-area (PPA) optimization [25], and the realization of sequential logic circuits such as an 8-bit shift register [26].

These preliminary studies demonstrated LLMs' ability in generating simple HDL codes for small atomic computation tasks. But signal-processing algorithms in wireless SDRs are far more complex than these simple computational tasks demonstrated in prior works. Realizing wireless signal-processing algorithms requires highly parallel processing (for real-time operations) and involves many complex mathematical computations.

Whether LLMs can successfully implement intricate signal-processing algorithms remains an open question. This thesis investigates the implementation of the FFT algorithm as a case study to assess the ability of LLMs in realizing intricate signal-processing algorithms for wireless communication. Two primary challenges are identified: 1) handling complex mathematical problems that require multi-step reasoning and 2) scheduling the execution of sub-modules within the complex hardware structure. Chapter 4 of this thesis explains these two challenges in details and provides our novel solutions to address these challenges.

1.5 Thesis Organization

The remainder of the thesis is organized as follows: Chapter 2 introduces the design and analysis of an IFDMA system with a novel FFT pruning scheme and an efficient scheduling algorithm for butterfly computation within the IFDMA FFT hardware realization, which contribute significantly to reduce the computation latency in an IFDMA system. In Chapter 3, this thesis addresses the packet-detection challenges in CSMA for enhanced system reliability. Chapter 4 investigates the feasibility of using LLMs to assist the implementation of advanced wireless communication systems on high-speed FPGA-based SDR platforms. Finally, Chapter 5 summarizes the thesis and discusses future works.

Chapter 2

Efficient FFT Computation in IFDMA

2.1 Background

Interleaved Frequency Division Multiple Access (IFDMA) is a broadband signal modulation and multiple-access technology for advanced wireless communications systems. Compared with other similar technologies [2, 3], such as Orthogonal FDMA (OFDMA), a salient feature of IFDMA is that its signal has a low Peak-to-Average Power Ratio (PAPR) [4]. The low PAPR of IFDMA brings about low signal distortion and high power efficiency when amplified by a power amplifier [5], making it a promising "green technology" for future wireless communications systems in which energy efficiency is a concern [6]. For the widespread deployment of IFDMA, however, the complexity of IFDMA transceivers needs to be minimized.

Recently, researcher in IFDMA have put forth a new class of IFDMA transceivers that is significantly less complex than conventional IFDMA transceivers [1, 7]. In essence, the design in [1, 7] allows a single IFFT/FFT module to perform the multiplexing/demultiplexing of multiple IFDMA data streams of different sizes, obviating the need to have IFFT/FFT modules of different sizes to cater to different IFDMA data streams. The key is to insert (for multiplexing at the transmitter) and extract (for demultiplexing at the receiver) the IFDMA data streams at different stages of the IFFT/FFT module according to their sizes.

In this thesis, we refer to the IFDMA transceiver design in [1, 7] as "compact IFDMA" to capture the fact that it performs IFDMA functions in a compact and efficient manner. And we refer to the specially designed IFFT/FFT in compact IFDMA as "compact IFDMA IFFT/FFT".

Although [1, 7] have laid down the theoretical foundation for compact IFDMA, its implementation in specific hardware (e.g., ASIC and FPGA) under resource constraints presents several technical challenges yet to be addressed. In [1, 7], the authors assumed perfect parallelism in the compact IFDMA FFT computation (i.e., unlimited number of processors). However, when the computation is to be implemented in an ASIC or FPGA, having abundant hardware to process the butterfly computations in parallel may not be possible, particularly when the number of subcarriers is large. Saving the die area is a vital issue in chip design [8], and the engineering design may be forced to use fewer processors for butterfly computations. Consequently, the amount of parallelism that can be achieved is limited, and scheduling the butterfly computation processes in compact IFDMA IFFT/FFT to minimize the computation latency is an issue.

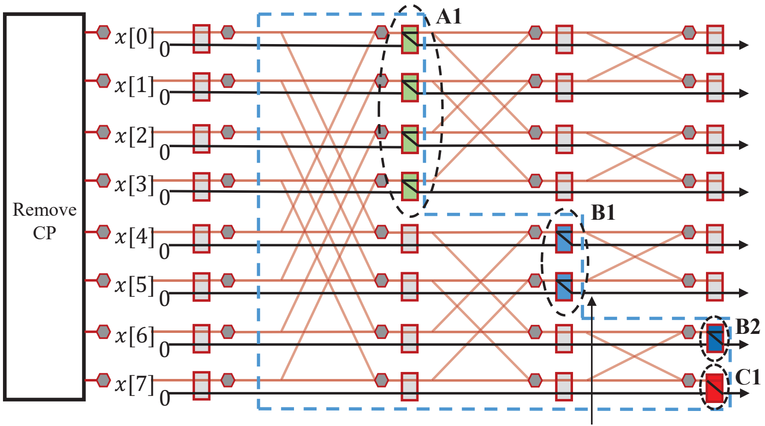
Although mature hardware implementations of conventional IFFT and FFT are available [27, 28, 29], they are not well matched to the compact IFDMA IFFT/FFT structure. As will be verified by the experimental results in Chapter 2.5.2, conventional IFFT/FFT implementations incur a much longer computation latency than the specially designed compact IFDMA IFFT/FFT in this thesis.

To achieve low latency, the proposed IFDMA IFFT/FFT implementation schedules the butterfly computations within an IFFT/FFT network by taking into account 1) the precedence relationships among the butterfly computations; and 2) the fact that not all butterfly computations within a full IFFT/FFT network need to be executed in the multiplexing and demultiplexing of IFDMA data streams.

This chapter focuses on the design of compact IFDMA FFT and skips the design of IFFT. The Cooley-Tukey FFT and IFFT have similar decomposition structures, and the design of compact IFDMA IFFT is similar to that of FFT. Interested readers are referred to Section IV and Secction V of [1] for details on the slight differences between compact IFDMA FFT and IFFT.

2.1.1 Quick Review and Motivation

Let us first briefly review how the compact IFDMA receiver works. Consider a scenario in which there are three users, A, B, and C, in an eight-subcarrier IFDMA system. Users A, B, and C require four subcarriers, three subcarriers, and one subcarrier, respectively. With the resource allocation scheme expounded in [7], a four-subcarrier IFDMA stream A1 is assigned to user A; a two-subcarrier IFDMA stream B1 plus a single-subcarrier IFDMA stream B2 are assigned to user B; and a single-subcarrier IFDMA stream C1 is assigned to user C. Fig. 2.1 shows the receiver's FFT [1] responsible for extracting/demultiplexing the respective IFDMA streams from the IFDMA signals.



The black arrow indicates a tap bus that can extract the wanted IFDMA data from different FFT stages.

Figure 2.1: An example on how the receiver extracts/demultiplexes the IFDMA data streams embedded in IFDMA signals.

As illustrated in Fig. 2.1, not all butterfly computations need to be executed. For example, data stream A1 can be extracted after the first-stage butterfly computations associated with it. The required data are extracted by four tap buses, bypassing the subsequent butterfly computations. Black arrows indicate the tap busses in Fig. 2.1 (details on the tap bus design can be found in Fig. 7 of [1]). Besides A1, Fig. 2.1 also shows the early extraction of B1. In general, only the IFDMA streams with one subcarrier need to go through the full FFT operations from the first stage to the last stage. In Fig. 2.1, all the butterfly computations outside the blue dashed line can be pruned.

In general, the butterfly computations that can be omitted depend on the users' subcarrier allocation. Fig. 2.2 shows another example, in which two users (A and B) are each allocated four subcarriers. During the operation of an IFDMA system, users may come and go, and their subcarrier demands may vary from time to time. This calls for a quick scheduling algorithm to determine the order of the butterfly computations in accordance with the active IFDMA subcarrier allocation at any moment in time, with the target of minimizing the computation latency.

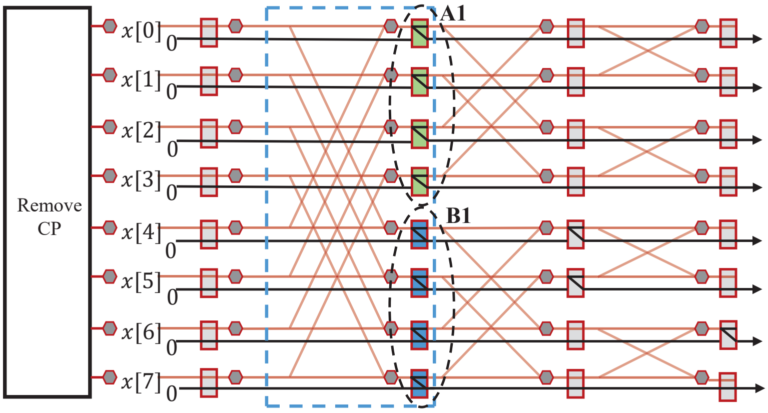


Figure 2.2: An example on a compact IFDMA FFT network with two users.

Conventional FFT implementations, however, are too rigid to serve the scheduling purpose in compact IFDMA FFT. Three conventional FFT implementations are the serial FFT implementation [27], the pipelined FFT implementation [28], and the parallel FFT implementation [29]. In these implementations, butterfly computations are executed in a fixed and rigid manner, i.e., from left to right, and from top to bottom. Although these fixed schedules work well in conventional FFTs that execute all butterfly computations, the pruned FFT structure resulting from compact IFDMA is less regular because the numbers of butterfly computations needed at different stages are different depending on the subcarrier allocations to the IFDMA data streams. The implication is that to fully exploit the potential efficiency that arises from the pruned butterfly computations, the remaining unpruned butterfly computations need to be scheduled carefully to reduce the time needed to finish the job. Otherwise, in the worst case, the time required would still be the same as that in the conventional FFT due to the precedence constraints among the remaining unpruned butterfly computations.

This chapter centers on a flexible computation scheduling algorithm tailored for IFDMA FFT. In general, task scheduling refers to the scheduling of a set of tasks with precedence relationships among them. A task cannot proceed before the inputs of the task are available, and the inputs of one task may be outputs of some preceding tasks. If we treat each butterfly computation as a task, it is then obvious that there are precedence relationships among the butterfly computations in the structure of the FFT. The precedence relationships can be captured by a directed acyclic graph (DAG) in which the vertexes are the tasks, and the directed edges indicate the precedence relationships between tasks. We refer to this special precedence graph as the FFT precedence graph.

In FPGA implementation, for example, we could design a customized butterfly computation processor and have a fixed number of such processors within the FPGA. The goal of the scheduling algorithm is to find an execution order for the tasks that can minimize the overall IFDMA FFT computation, subject to constraints of the precedence relationships and the number of processors.

The general scheduling problem is NP-complete. However, we will provide evidence that the scheduling algorithm for IFDMA FFT is near-optimal. The algorithm 1) works well with arbitrary numbers of processors; 2) works well for random subcarrier allocations; and 3) has much lower computation time than conventional FFT implementations.

2.1.2 Related Works

Task scheduling subject to precedence constraints is tough in general. Karp first studied this problem in his seminal paper [30], where the task scheduling problem is reduced to a 3-satisfiability (3SAT) problem and proved to be one of the first 21 NP-complete problems. Subsequently, [31] found that task scheduling to minimize the computation time subject to a fixed number of processors and a uniform task processing time is also NP-complete. The scheduling problem in this thesis, unlike the general scheduling problem, is specific to the FFT precedence graph resulting from the IFDMA pruned FFT.

The first category of prior works concerns task scheduling in conventional unpruned FFT with the objective of computation speedup over specialized hardware. For example, [32] improved the butterfly task scheduling in a very-long-instruction-word (VLIW) digital signal processors (DSP) chip using a software pipelining technique called modulo scheduling. This scheduling algorithm exploits the instruction-level parallelism (ILP) feature in the VLIW DSP platform to schedule multiple loop iterations in an overlapping manner [33]. In [34, 35], the authors put forth the celebrated Fastest Fourier Transform in the West (FFTW) C library for the FFT implementation on multiprocessor computers. Other works like UHFFT [36] and SPIRAL [37] also contributed to the FFT implementation on multiprocessor computers and DSP chips. What distinguishes this work from the above works is that the design here considers pruned FFT arising from compact IFDMA.

Another category of prior works concerns task scheduling in pruned FFT [38, 39, 40, 41, 42], but not that arising from compact IFDMA. A key difference that distinguishes this work from these prior works is that for IFDMA, we want selected intermediate data at various stages inside the FFT flow graph. These intermediate data extractions, and the grouping of the intermediate data, bring forth a completely new structure for the pruned FFT.

Three types of conventional FFT pruning schemes were studied in previous works. The first is input-pruned FFT [38], where some computations can be eliminated when the FFT inputs have many zeros. The second is output-pruned FFT [39], where some computations can be eliminated when some of the FFT outputs are not needed. The third combines the aforementioned input pruning and output pruning [40]. In short, the above FFT pruning works take away the inputs with no data and take away the outputs not required. In [41], the authors studied the scheduling problem when many FFT inputs are zero. In [42] , the authors investigated the efficient task scheduling in OFDMA applications, where only a small number of the FFT outputs are wanted by a user.

In the FFT implementation for IFDMA applications, however, we want to extract selected data at various stages within the FFT flow graph. In the context of the complete FFT computation, these are intermediate computation results rather than the final outputs. For example, in Fig. 2.1, four intermediate outputs of the first-stage butterfly computations are wanted for IFDMA stream A1. That leads to a very different structure than pruned graphs studied by prior works. The new FFT pruning structure calls for new scheduling algorithms that can exploit the particular structure induced by IFDMA.

2.1.3 Contributions and Findings

We summarize the thesis's contributions in this chapter as follows. To begin with, we developed a new type of FFT pruning scheme tailored to IFDMA transceivers. Further, we put forth an algorithm, referred to as multi-priority scheduling (MPS), to find a optimal task execution schedule for compact IFDMA FFT in terms of computation time.

To establish the near optimality of MPS, we derive an analytical lower bound for the computation time and show that MPS can almost always find schedules that meet the lower bound. We refer to the FFT computation scheduled by MPS as MPS-FFT. We conducted large-scale experiments to comprehensively study the performance of MPS-FFT with up to 1024 subcarriers. Two key experimental results of MPS-FFT are as follows:

1) When the number of processors is a power of two, MPS-FFT subject to a randomly selected IFDMA subcarrier allocation can reach the computation-time lower bound with probability approaching one. Quantitatively, the probability is larger than $\sqrt[\xi_0+1]{0.05}$, where ξ_0 is more than 2.5 million.

2) When the number of processors is arbitrary, MPS-FFT subject to a randomly selected IFDMA subcarrier allocation can reach the computation-time lower bound with probability of at least 98.70%. In the few cases that fail to reach the lower bound, the computation time is at most 6% larger than the lower bound.

Experiments also demonstrate that MPS-FFT has a much shorter computation time than the conventional FFT schemes do. The speedup achieved by MPS-FFT can be attributed to two factors: (i) the pruning scheme of MPS-FFT reduces more than 11.21% of the butterfly computations on average; (ii) MPS-FFT of 64-point or larger sizes have a more than 98.42% processor utilization rate when the number of processors is a power-of-two, almost twice higher than conventional FFT implementations with the same number of processors. Thanks to the above two benefits, the computation time of MPS-FFT with a number of power-of-two processors reduces the computation time of conventional FFTs with the same number of processors by 59.86% on average.

2.2 Definitions and Problem Formulation

2.2.1 Problem Formulation

We consider an N-point FFT in a compact IFDMA receiver. In the FFT network, there are $n = \log_2 N$ FFT stages, with each stage having 2^{n-1} butterfly tasks.

For IFDMA subcarrier allocation, it is convenient to first map subcarriers to "bins" through a bit-reversal mapping so that the subcarrier indexes, after bit reversal, become the bin indexes, and subcarrier allocation becomes bin allocation (see Section III of [7] on bit-reverse index mapping of subcarriers to bins). This mapping process allows an IFDMA stream to be allocated consecutive bins that correspond to regularly-interspersed subcarriers, as is needed for the IFDMA stream.

A bin allocation can be expressed by an ordered list $\{S_0, S_1, ...S_{R-1}\}$, where R is the number of IFDMA streams, and S_r $(r = 0, ..., R-1)$ is the set of bins allocated to IFDMA stream r. Fig. 2.3 gives an example of the construction of the FFT precedence graph given an IFDMA bin allocation. Fig. 2.3(a) shows a specific bin allocation for a system with 16 subcarriers and 11 IFDMA streams. IFDMA stream 0 is allocated the bins $S_0 = \{0, 1, 2, 3\}$, IFDMA stream 1 is allocated the bins $S_1 = \{4, 5\}$, and so on. Fig. 2.3(b) shows the corresponding IFDMA FFT network for the receiver, where the green boxes are the places from which the data of the IFDMA streams are extracted. Fig. 2.3(c) shows the corresponding FFT precedence graph. A vertex in the precedence graph corresponds to a 2×2 butterfly computation in the FFT network.

We next explain how the precedence graph in Fig. 2.3(c) is obtained. Consider stage i $(i = 0, 1, ..., \log_2 N - 1)$ of the FFT network. We denote task j $(j = 0, 1, ..., N/2 - 1)$ and FFT output k $(k = 0, 1, ...N - 1)$ at stage i by $v_{i,j}$ and $d_{i,k}$, respectively. With respect to Fig. 2.3(b), the butterfly computation in blue corresponds to task 6 (enumerating from top to bottom) at stage 2 (enumerating from left to right). Thus, the task is written as $v_{2,6}$, and the outputs of this task

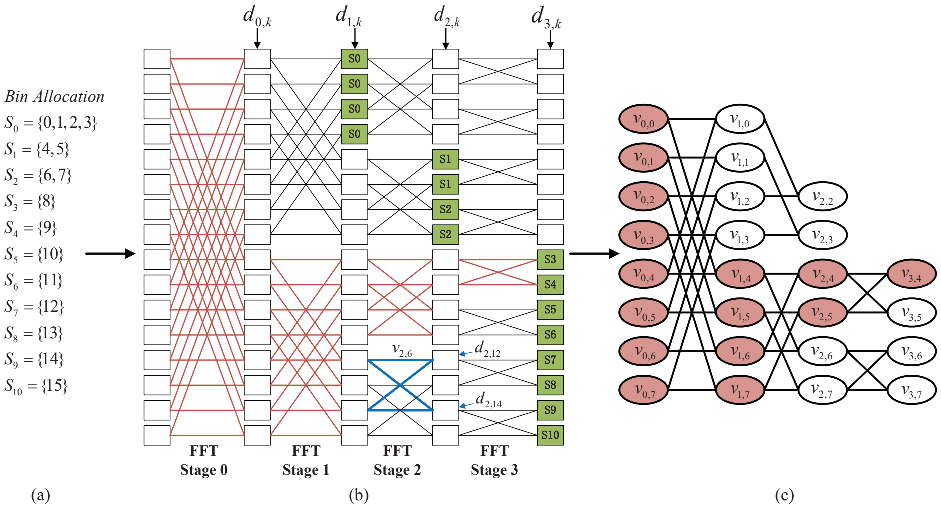


Figure 2.3: An example to illustrate how to obtain an FFT precedence graph given $\{S_0, S_1, ...S_{R-1}\}$.

are written as $d_{2,12}$ and $d_{2,14}$. The index k in $d_{i,k}$ is labeled from top to bottom, with $k = 0$ corresponding to a rectangular box at the top and $k = 15$ corresponding to a rectangular box at the bottom.

Let D denote the set of desired FFT outputs of all IFDMA streams, i.e., green boxes in Fig. 2.3(b). Further, let V denote the set of tasks that need to be executed to obtain these desired FFT outputs. The precedence graph in Fig. 2.3(c) is obtained as follows.

In Fig. 2.3(b), to obtain output $d_{3,8}$ required by IFDMA stream S_3, task $v_{3,4}$ needs to be completed. However, before task $v_{3,4}$ can be executed, tasks $v_{2,4}$ and $v_{2,5}$ need to be executed first, because their outputs $d_{2,8}$ and $d_{2,9}$ are inputs to $v_{3,4}$. Further, before $v_{2,4}$ and $v_{2,5}$ can be executed, some tasks in the prior stage need to be executed, and so on and so forth. In Fig. 2.3(b), we mark all of the butterflies that need to be executed to obtain $d_{3,8}$ in pink. The vertexes in Fig. 2.3(c) that correspond to these butterfly tasks are also pink. By tracing backward from the desired IFDMA stream outputs, we can obtain all the tasks that need to be executed and the precedence relationships between them to arrive at the overall precedence graph.

We note that every task except those at stage 0 requires the outputs of two specific tasks in the prior stage as its inputs, and the task cannot be executed unless these two preceding tasks are completed first. We define the concept of parent vertex and child vertex as follows:

Definition 1 (Parent Vertex and Child Vertex)**.** *If* v_{i+1,j_0} *cannot be executed unless* v_{i,j_1} *and* v_{i,j_2} *are completed, we refer to* v_{i+1,j_0} *as a child of* v_{i,j_1} *and* v_{i,j_2}*. Tasks* v_{i,j_1} *and* v_{i,j_2} *are referred to as parents of* v_{i+1,j_0}*.*

Formally, we define the FFT precedence graph together with its associated notations as follows:

Definition 2 (FFT Precedence Graph)**.** *A precedence graph* $G(V,E)$ *describes the task dependencies between a set of tasks, where*

1) Vertex $v_{i,j} \in V$ *represents task* (i,j) *within* V*.*

2) There is an edge $(v_{i,j_1} \to v_{i+1,j_2}) \in E$ *if task* v_{i,j_1} *must be executed before task* v_{i+1,j_2} *can be executed, i.e.,* v_{i,j_1} *is a parent of* v_{i+1,j_2}*.*

Since butterfly tasks are executed by processors with the same capacity, the processing times of tasks are the same (i.e., each vertex in the precedence graph takes the same amount of time to execute). We refer to the execution time of a task as one time slot.

With the above backdrop, we define the compact IFDMA FFT scheduling problem as follows:

Definition 3 (The FFT Scheduling Problem in Compact IFDMA)**.** *Given*

1) an FFT precedence graph $G(V,E)$ *associated with a set of IFDMA streams together with their bin allocations, and*

2) M *processors, each of which can complete one task in exactly one time slot,*

The problem is to determine a task-execution schedule $\mathbf{X} = (\chi_0, ..., \chi_{T-1})$*, where* χ_t *is the set of tasks to be executed in time slot* t*, and* T *is the completion time of the collection of tasks, such that,* *1) the precedence relationships are satisfied;*

2) there are no more than M tasks in each of χ_t, and $\bigcup_t \chi_t$ consists of all the tasks in the precedence graph;

3) T is minimized.

Henceforth, unless stated otherwise, we refer to the above FFT scheduling problem in compact IFDMA systems simply as the "scheduling problem". We refer to an algorithm to solve the scheduling problem as a "scheduling algorithm".

Remark 1. *The schedule $\mathbf{X} = (\chi_0, ..., \chi_{T-1})$, once determined, may be reused for a duration of time. The reason is as follows. In many applications, the set of IFDMA streams remains the same for a duration of time. As long as the bin allocations do not change, the subcarriers used by each of the streams remain the same. Each stream uses the same subcarriers to transmit its successive IFDMA symbols. An IFDMA FFT needs to be computed for each symbol period for the collection of IFDMA streams. However, the precedence graph remains the same for the successive FFTs. Thus, once we obtain schedule $\mathbf{X}$, the same schedule can be used to perform the IFDMA FFT for the successive IFDMA symbols. In particular, there is no need to compute $\mathbf{X}$ and build new FFT precedence graphs repeatedly. If the set of IFDMA streams changes (e.g., an IFDMA stream ends and leaves, or a new IFDMA stream arrives to join the existing IFDMA stream), a new schedule $\mathbf{X}$ will need to be computed.*

Remark 2. *We make two practical assumptions by transforming the efficient execution of FFT computation within an IFDMA transceiver into the problem formulation above. First, we assume that all subcarriers are fully loaded in the IFDMA system, i.e., all bins are allocated to a user. This assumption simplifies the construction of the FFT precedence graph, and it is practical since the service provider can always use zero padding if the IFDMA system is not fully loaded. To remove this assumption, the process of constructing the FFT precedence graph,* *illustrated in Fig. 2.3, would need to be revised. However, this does not affect our scheduling algorithm, as it can take any feasible FFT precedence graph as input. In other words, removing the "fully loaded" assumption does not impact the major contribution of this paper – it only requires an additional steps in the transformation between user bin allocation and the FFT precedence graph. Moreover, if the fully loaded assumption were removed, we could expect a lower computation latency in general, given that more butterfly computations could be pruned with the non-fully-loaded setup. Second, regarding the transceiver implementation on FPGA or ASIC, we assume that the FFT computation in compact IFDMA is "computation bounded" rather than "communication bounded" . This implies that the latency caused by "sending the output of one butterfly computation from one processor to another processor as input" is significantly lower than the time required to perform the butterfly computation itself. This assumption allows us to focus only on the computational aspect within the chip. It is highly practical, considering that internal buses within an FPGA/ASIC chip provide a communication bandwidth exceeding 10 Gbps [43], while the output of a butterfly computation is just two complex numbers requiring only several bytes for representation, taking several nanoseconds for transmission between different processors only. The time consumption of a butterfly computation, on the other hand, takes significantly longer time (see Appendix E for detailed analysis, using an eight-point Xilinx FFT IP core as an example). Given this background, the "computation bounded" assumption is both practical and reasonable.*

2.2.2 Feasibility of a Task-execution Schedule

We next describe a step-by-step procedure to check whether a given task-execution schedule $\mathbf{X} = \{\chi_0, ..., \chi_{T-1}\}$ is feasible according to criteria 1) and 2) in Definition 3. The propsoed scheduling algorithm (to be described later) follows this step-by-step procedure to construct the schedule to make sure it adheres to the feasibility criteria in each step of the way. In essence, the step-by-step procedure starts with χ_0 and proceed progressively to $\chi_1, \chi_2, ...$, updating the precedence graph along the way to make sure that the precedence relationships are obeyed each step of the way.

Consider the beginning of time slot t, the vertexes in χ_t must have no parents and there are no more than M tasks in χ_t. If this is not satisfied, then the schedule $\mathbf{X}$ is not feasible. If χ_t passes the test, we then remove the vertexes in χ_t together with their output edges from the precedence graph. The updated precedence graph is used to check the feasibility of χ_{t+1}. We say that $\mathbf{X}$ is a feasible schedule if and only if the precedence graph is nil after vertex removal in time slot $T-1$.

2.2.3 Optimality of a Heuristic Scheduling Algorithm

Given a feasible schedule $\mathbf{X}$, the next question one may ask is whether the schedule is optimal in that the overall computation time T is minimized. Recall from the introduction that the scheduling problems associated with the general non-FFT precedence graph have been proven to be NP-complete [30, 31]. Designing good heuristic algorithms for them with provable bounds is challenging.

The FFT precedence graphs, however, are specific precedence graphs with a regular structure. Chapter 2.3.2 gives a heuristic algorithm that solves the FFT scheduling problem to produce a schedule $\mathbf{X} = \{\chi_0, ..., \chi_{T-1}\}$. Chapter 2.4 gives a lower bound on T (denoted by T^L) for a given FFT precedence graph. Chapter 2.5.2 shows by experiments that the proposed scheduling algorithm, MPS, when given a randomly selected subcarrier allocation instance (note: each subcarrier allocation instance can be mapped to a corresponding FFT precedence graph), can find a schedule that can reach the computation-time lower bound with very high probability.

We characterize the optimality of a scheduling algorithm by a duple $(\eta_{n,M}, \gamma_{n,M})$ defined as follows. For a given FFT scheduling problem with 2^n subcarriers and M processors, if a scheduling algorithm produces a feasible schedule $\mathbf{X}$ with $T = T^L$, then the schedule is optimal. Among all subcarrier allocation instances tested, we denote the percentage of schedules produced by a heuristic algorithm that satisfies $T = T^L$ by $\eta_{n,M}$. Specifically,

$$\eta_{n,M} = E\left[Pr(T = T^L)\right] \tag{2.1}$$

where $Pr(\cdot)$ is the probability of an event, and $E[\cdot]$ is the expectation of a random variable. The probability of finding the optimal schedule for a random subcarrier allocation instance is no less than $\eta_{n,M}$ (note that $T > T^L$ does not mean the schedule is not optimal, since T^L is just a lower bound).

For the remaining $1 - \eta_{n,M}$ fraction of the schedules with $T > T^L$, the gap between T and the best possible computation time T^* is $T - T^* \leq T - T^L$. The percentage gap between T and T^* is $(T - T^*)/T^* < \left(T - T^L\right)/T^L$. We define

$$\gamma_{n,M} = E\left[\left(T - T^L\right)/T^L\ \right] \tag{2.2}$$

to be the upper bound of the expected percentage gap, i.e., the average percentage gap between T and T^* is no more than $\gamma_{n,M}$.

The input to a scheduling algorithm is an FFT precedence graph. As far as scheduling algorithms are concerned, FFT precedence graphs that are isomorphic to each other are equivalent. If a scheduling algorithm finds the optimal schedule for an FFT precedence graph, the optimal scheduling also applies to an isomorphic FFT precedence graph after an isomorphic transformation. Appendix A delves into the isomorphism of FFT precedence graphs.

When running experiments over scheduling algorithms by subjecting them to different FFT precedence graphs, it would be desirable to remove isomorphism so that graphs with many isomorphic instances are not over-represented.

We denote the complete set of FFT precedence graphs by F_n, wherein all elements are non-isomorphic. The cardinality of F_n is written as f_n. We refer to the bin allocations that lead to isomorphic FFT precedence graphs as isomorphic bin allocations. Among isomorphic bin allocations, we only select one to put into F_n. We refer the reader to Appendix A on how we do so.

Appendix B explains the principle to generate the complete set of non-isomorphic instances F_n for various n. When f_n is not large, it is not an issue to use F_n, the complete set, as the test set to conduct experiments and obtain $\eta_{n,M}$ and $\gamma_{n,M}$. However, as shown in Appendix B, f_n becomes prohibitively large when n is more than 6. Table 2.1 lists the f_n values for various n, which are also derived in Appendix B.

Table 2.1: Number of non-isomorphic instances for various n

n	N	f_n	n	N	f_n
1	2	2	6	64	2598061
2	4	4	7	128	$3.3750E + 12$
3	8	11	8	256	$5.6952E + 24$
4	16	67	9	512	$1.6218E + 49$
5	32	2279	10	1024	$1.3151E + 98$

As shown, when $n \geq 7$, the number of instances in F_n is so large that it is impractical to generate all the instances and use the complete set of non-isomorphic instances as the test set. Hence, for $n \geq 7$, instead of the full set F_n, we randomly generate a subset of F_n (referred to as $F_n^{'}$) to serve as the test set. Appendix C explains how we randomly generate the subset of non-isomorphic instances for $n \geq 7$.

For $n \geq 7$, we can perform tests on $F_n^{'}$ and characterize $(\eta_{n,M}, \gamma_{n,M})$ statistically. We first consider $\eta_{n,M}$. For a 2^n-point IFDMA FFT with M available processors, we randomly sample ξ_0 instances with replacement from F_n. Define Y_{ξ_1} to be the event that ξ_1 of the ξ_0 instances are instances for which the heuristic algorithm can find a solution reaching the lower bound. The conditional probability of Y_{ξ_1} given $\eta_{n,M}$ is

$$P(Y_{\xi_1}|\eta_{n,M}) = \begin{pmatrix}\xi_0\\ \xi_1\end{pmatrix} \eta_{n,M}^{\xi_1}(1-\eta_{n,M})^{\xi_0-\xi_1}. \tag{2.3}$$

We are interested in the *a posteriori* probability density function (PDF) $p(\eta_{n,M}|\,Y_{\xi_1})$. We have, by Bayes' rule,

$$p(\eta_{n,M}|Y_{\xi_1}) = \frac{p(\eta_{n,M})\cdot P(Y_{\xi_1}|\eta_{n,M})}{P(Y_{\xi_1})}. \tag{2.4}$$

However, we do not know the *a priori* PDF $p(\eta_{n,M})$. As a conservative measure, we can assume the worst case of having no knowledge on $p(\eta_{n,M})$ and let $p(\eta_{n,M})=1$, $\forall\eta\in[0,1]$. Thus,

$$p(\eta_{n,M}|Y_{\xi_1}) = \frac{P(Y_{\xi_1}|\eta_{n,M})}{P(Y_{\xi_1})}. \tag{2.5}$$

As a probability density, $p(\eta_{n,M}|Y_{\xi_1})$ must integrate to 1. That is to say, $\int p(\eta_{n,M}|Y_{\xi_1})d\eta_{n,M}=1$, which gives,

$$\int \tfrac{P(Y_{\xi_1}|\eta_{n,M})}{P(Y_{\xi_1})}d\eta_{n,M} = 1 \Leftrightarrow P(Y_{\xi_1}) = \int P(Y_{\xi_1}|\eta_{n,M})\,d\eta_{n,M}\ . \tag{2.6}$$

Thus, we have

$$\begin{aligned} p(\eta_{n,M}|Y_{\xi_1}) &= \frac{P(Y_{\xi_1}|\eta_{n,M})}{\int_0^1 P(Y_{\xi_1}|\eta_{n,M})d\eta_{n,M}} \\ &= \frac{\eta_{n,M}^{\xi_1}(1-\eta_{n,M})^{\xi_0-\xi_1}}{\int_0^1 \eta_{n,M}^{\xi_1}(1-\eta_{n,M})^{\xi_0-\xi_1}d\eta_{n,M}}, \\ &= \frac{\eta_{n,M}^{\xi_1}(1-\eta_{n,M})^{\xi_0-\xi_1}}{B\left(\xi_1+1,\xi_0-\xi_1+1\right)} \end{aligned} \tag{2.7}$$

where $B\left(m,n\right)=\int_0^1 x^{m-1}(1-x)^{n-1}dx$ is the well-known Beta function[1].

Let α be the target confidence, let $\eta_{n,M}^L$ be the lower bound of $\eta_{n,M}$, and let $\int_{\eta_{n,M}^L}^1 p(\eta_{n,M}|Y_{\xi_1})\ d\eta_{n,M}=\alpha$. We then have

$$\begin{aligned} \tfrac{\int_{\eta_{n,M}^L}^1 \left(\eta_{n,M}\right)^{\xi_1}(1-\eta_{n,M})^{\xi_0-\xi_1}\ d\eta_{n,M}}{B(\xi_1+1,\xi_0-\xi_1+1)} &= \alpha \\ \Leftrightarrow I_{\eta_{n,M}^L}(\xi_1+1,\xi_0-\xi_1+1) &= 1-\alpha \end{aligned}, \tag{2.8}$$

---

[1] The following mathematical properties of the Beta function, regularized Beta function, and incomplete Beta function can be found in Section 6.2 and Section 26.5 of [44]

where

$$I_{\eta^L_{n,M}}\left(\xi_1+1,\xi_0-\xi_1+1\right)=\frac{B\left(\eta^L_{n,M},\xi_1+1,\xi_0-\xi_1+1\right)}{B\left(\xi_1+1,\xi_0-\xi_1+1\right)} \tag{2.9}$$

is the regularized Beta function, and

$$B(\eta^L_{n,M},\xi_1+1,\xi_0-\xi_1+1)=\int_0^{\eta^L_{n,M}} x^{\xi_1+1}(1-x)^{\xi_0-\xi_1+1}dx \tag{2.10}$$

is the incomplete Beta function.

With partial integration, $I_{\eta^L_{n,M}}\left(\xi_1+1,\xi_0-\xi_1+1\right)$ can be further written as,

$$I_{\eta^L_{n,M}}\left(\xi_1+1,\xi_0-\xi_1+1\right)=\sum_{j=\xi_1+1}^{\xi_0+1}\tfrac{(\xi_0+1)!}{j!(\xi_0+1-j)!}\left(\eta^L_{n,M}\right)^j\left(1-\eta^L_{n,M}\right)^{\xi_0+1-j}. \tag{2.11}$$

We obtain $\eta^L_{n,M}$ by applying (2.11) to the incomplete Beta function in (2.8).

We next consider how to obtain the α-percentile confidence interval of $\gamma_{n,M}$ for $n\geq 7$. For the $\xi_0-\xi_1$ non-lower-bound-reaching schedules (if any), we denote the mean and variance of the observed percentage gap $\left(T-T^L\right)/T^L$ by μ and σ^2, respectively. With the central-limit theorem [45], we construct a random variable Z:

$$Z=\frac{\mu-\gamma_{n,M}}{\sigma/\sqrt{\xi_0-\xi_1}}. \tag{2.12}$$

With confidence target α and critical z-score $z_{1-\alpha/2}$, we have

$$P(-z_{1-\alpha/2}\leq Z\leq z_{1-\alpha/2})=\alpha. \tag{2.13}$$

Applying (2.12) to (2.13), we then say,

$$\mu-z_{1-\alpha/2}\cdot\frac{\sigma}{\sqrt{\xi_0-\xi_1}}\leq\gamma_{n,M}\leq\mu+z_{1-\alpha/2}\cdot\frac{\sigma}{\sqrt{\xi_0-\xi_1}} \tag{2.14}$$

with confidence α.

2.3 Multi-Priority Scheduling for IFDMA FFT

We now delve into the structural properties of the FFT precedence graph. A heuristic algorithm tailored to the FFT scheduling problem is developed by leveraging the structural properties of IFDMA FFT.

2.3.1 Structural Properties of FFT Precedence Graphs

Several important structural properties of FFT precedence graphs are given below:

- **Property 1** (Number of Parents): Vertexes at stage 0 have no parent. Vertexes at other stages have two parents located at the preceding stage.

- **Property 2** (Number of Children): Vertexes at stage $\log_2 N - 1$ (i.e., last stage) have no children. Vertexes at other stages can have one, two, or no children. In Fig. 2.3(c), for example, $v_{3,4}$ at the last stage has no children, while $v_{1,0}$ has one child, $v_{1,4}$ has two children, and $v_{2,2}$ has no children.

- **Property 3** (Parents): Consider vertex $v_{i,j}$ at stage $i(1 \leq i \leq \log_2 N - 1)$. Its two parents at stage $i-1$ can be identified as follows:

  1) If $j \bmod \left(2^{\log_2 N - i}\right) < 2^{\log_2 N - i - 1}$, the two parents are $v_{i-1,j}$ and $v_{i-1,j+2^{\log_2 N - i - 1}}$.
  2) If $j \bmod \left(2^{\log_2 N - i}\right) \geq 2^{\log_2 N - i - 1}$, the two parents are $v_{i-1,j-2^{\log_2 N - i - 1}}$ and $v_{i-1,j}$.

- **Property 4** (Children): Consider $v_{i,j}$ at stage $i(0 \leq i \leq \log_2 N - 2)$. Its children at stage $i+1$ (if any) can be identified as follows:

  1) If $j \bmod \left(2^{\log_2 N - i - 1}\right) < 2^{\log_2 N - i - 2}$, the potential children are $v_{i+1,j}$ and $v_{i+1,j+2^{\log_2 N - i - 2}}$.
  2) If $j \bmod \left(2^{\log_2 N - i - 1}\right) \geq 2^{\log_2 N - i - 2}$, the potential children are $v_{i+1,j-2^{\log_2 N - i - 2}}$ and $v_{i+1,j}$.

Note that one or both of the potential children may be absent in the FFT precedence graph. The above properties lead to a obvious observation that the FFT butterfly structure in turn induces a certain partial butterfly pattern in the precedence graph. This is, if v_{i,j_1} has two children, there must be another vertex v_{i,j_2} sharing the same two children vertexes with v_{i,j_1}. If v_{i,j_1} has only one child, obviously, there is another parent at stage i sharing the same child with v_{i,j_1}.

With the above observations, we give the definitions of "vertex pair" and "the companion of a vertex" below.

Definition 4 (Vertex Pair and the Companion of a Vertex)**.** *If a parent vertex shares children with another parent vertex, we refer to the two parent vertexes as a vertex pair. Each of the parent vertexes is the companion of the other.*

Fig. 2.4 illustrates the legitimate parent-child structures within the FFT precedence graph. A vertex can have zero, one or two children. Fig. 2.4(a) shows the case of two vertexes sharing two common children. Fig. 2.4(b) and (c) show the cases of two vertexes, each having one child, and that they share this common child.

Note that the structures in Fig. 2.4 are only applicable to the initial FFT precedence graph before the scheduling algorithm is executed. This is because if a parent vertex is selected for execution and then removed from the FFT precedence graph, the structure of the graph changes. Vertex pairs and the companion relationship in Definition 4, on the other hand, are applicable to the precedence graph throughout the execution of the scheduling algorithm. If vertex A is the companion of vertex B, and it is selected for execution before vertex B, we refer to vertex A as a selected companion of vertex B.

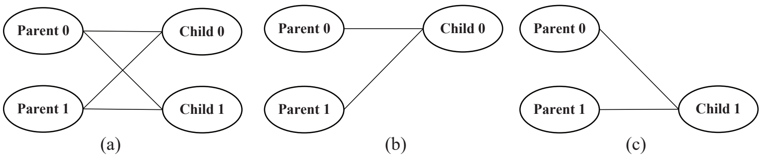


Figure 2.4: Legitimate parent-child structures within the FFT precedence graph.

2.3.2 The Heuristic Scheduling Algorithm

We now construct a heuristic scheduling algorithm that exploits the structure of the FFT precedence graph. Fig. 2.5 presents an example of the evolution of the FFT precedence graph during the execution of the heuristic algorithm. The example assumes the same IFDMA bin allocation as in Fig.

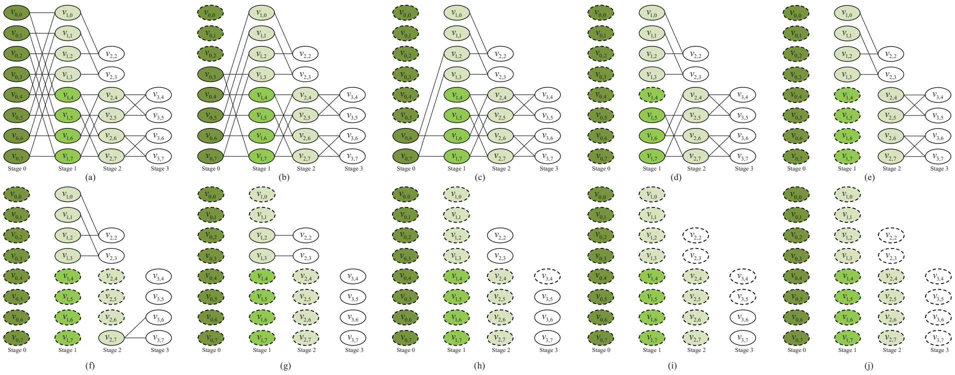


Figure 2.5: Evolution of the FFT precedence graph during the execution of the heuristic algorithm, assuming three processors. The vertexes enclosed by a solid boundary belong to the precedence graph. The dashed vertexes do not belong to the precedence graph, and they are shown here for illustrative purposes only.

2.3(a) of Chapter 2.2.3. The vertexes are marked in different colors to facilitate later explanation of the heuristic algorithm. Here, we assume that the number of processors is three. A scheduling algorithm begins with the initial precedence graph shown in Fig. 2.5(a) and ends with the final precedence graph shown in Fig. 2.5(j). Each of the subfigures from Fig. 2.5(b) to Fig. 2.5(i) shows an intermediate FFT precedence graph at the end of a time slot. When a vertex is removed from the FFT precedence graph, we mark it as a dashed ellipse, the removed edges are simply not shown.

We refer to the scheduling algorithm as multi-priority scheduling (MPS). As the name suggests, MPS associates each vertex (task) with a priority vector $\langle P_1, P_2, ..., P_H \rangle$ to determine its execution priority. In particular, the priority element P_i takes precedence over the priority element P_{i+1} when comparing the priorities of two vertexes. For example, if the $P_1, P_2, ..., P_{i-1}$ of tasks A and task B are the same, but P_i of task A is higher than of task B, then task A is of higher priority than task B.

Throughout this chapter, we study the MPS algorithm for which the priority vector $\langle P_1, P_2, P_3, P_4 \rangle$ has four elements. This algorithm was applied in the example of Fig. 2.5. Let us now specify the elements in $\langle P_1, P_2, P_3, P_4 \rangle$ and explain how they are used to yield the scheduling results in Fig. 2.5.

First, P_1 of a vertex is the number of generations of descendants it has, wherein "descendants" refers to its children, its children's children, and so on. Fig. 2.5(a) illustrates how we count the number of generations. We mark the vertexes in Fig. 2.5(a) with four different colors according to how many generations of descendants they have. Vertexes with no child are colored in white. Their parents have one generation of descendants, and they are colored in light green. Vertexes in green have two generations of descendants, and vertexes in dark green have three generations of descendants.

The intuition for setting P_1 as above is as follows. Recall that the children of a vertex must be executed in a different time slot than the vertex. Thus, the number of generations of descendants of a vertex corresponds to the number of extra time slots that are needed in addition to the time slot used to execute the vertex. It will be advantageous to select the vertex with more generations of descendants to free up the dependencies of its descendants on it.

Let us look at Fig. 2.5. The precedence graph after time slot 2 is shown in Fig. 2.5(c). Three tasks can be selected for execution in time slot 3. Among the eight executable vertexes $(v_{0,6}, v_{0,7}, v_{1,0}, ...v_{1,5})$ in Fig. 2.5(c), MPS selects all of the vertexes with high P_1 value $(P_1(v_{0,6}) = P_1(v_{0,7}) = 3)$ and one of the vertexes with medium P_1 value $(P_1(v_{1,4}) = 2)$. The algorithm does not select the light green vertexes in time slot 3 because their P_1 is lower than those in green and dark green.

Next, we present the specification of P_2. We divide the vertexes into three types. Childless type refers to vertexes with no children. Paired type refers to vertexes for which both the vertex itself and its companion have not yet been selected for execution (see Definition 4 for the concept of companion). Singleton type refers to unselected vertexes whose companion has already been selected for execution in a prior time slot. Accordingly, P_2 is set to 0, 1, and 2, respectively.

The intuition for setting P_2 as above is as follows. Since the selection of childless vertexes brings no new executable vertex in the next time slot, we give it the lowest P_2. For paired vertexes, it takes two processors to execute them together in the same time slot and obtain new executable vertex(es) (i.e., the children) in the next slot. For singleton vertexes, it takes only one processor to obtain the same number of new executable vertex(es) in the next slot. So, we give the middle and the highest P_2 to paired vertex and singleton vertex, respectively. Note that P_2 is a myopic measure while P_1 looks further ahead. Hence, P_1 takes precedence over P_2.

Let us look at the example of Fig. 2.5 again. The precedence graph at the end of time slot 3 is shown in Fig. 2.5(d). Both $v_{1,5}$, $v_{1,6}$ and $v_{1,7}$ have the highest P_1 among the unselected vertexes. Note that $v_{1,5}$ and $v_{1,7}$ are of paired type, because they form a pair and neither of them has been selected for execution, while $v_{1,6}$ is of singleton type because its companion $v_{1,4}$ has been selected for execution in a previous time slot. Hence, the selection order of the following time slot (time slot 4) is $v_{1,6} \rightarrow v_{1,5} \rightarrow v_{1,7}$. If M were to be 2, then only $v_{1,6}$ and one of $v_{1,5}$ or $v_{1,7}$ would be selected; however, since $M = 3$ in the example, all of them are selected.

Next, P_3 of a vertex is the number of children it has. Note that both a singleton-type vertex and a pair of paired-type vertexes may have two children in the next time slot. But an intelligent algorithm should select the former for execution first. Hence, P_2 takes precedence over P_3.

In Fig. 2.5, the precedence graph at the end of time slot 5 is shown in Fig. 2.5(e). All vertexes in light green are executable, and they have the same P_1 and P_2 values. The MPS algorithm prefers $v_{2,4}$,$v_{2,5}$, $v_{2,6}$ and $v_{2,7}$, because each of them has two children, while each of the other four executable vertexes ($v_{1,0}$,$v_{1,1}$, $v_{1,2}$ and $v_{1,3}$) has only one child. Since there are only three processors, the algorithm selects three vertexes among $v_{2,4}$,$v_{2,5}$, $v_{2,6}$ and $v_{2,7}$. Hence, the selection order of the following time slot (time slot 6) is $v_{2,4} \rightarrow v_{2,5} \rightarrow v_{2,6}$.

Next, if executable vertexes have the same P_1, P_2 and P_3 values, we can randomly select the vertexes to execute. However, we are interested in a deterministic algorithm (for analytical purposes to establish a performance bound, as will be elaborated later). Thus, we have P_4 to break the tie. We select those vertexes from the top of the precedence graph to the bottom of the precedence graph. So, we assign P_4 to a vertex according to the row of the precedence graph at which it is located. Specifically, $P_4 = (N/2\ -1) - j$, where j is the row that $v_{i,j}$ lies in.

In Fig. 2.5, the precedence graph at the end of time slot 7 is shown in Fig. 2.5(h). There are five executable vertexes with the same P_1, P_2 and P_3 values. Among them, $v_{2,2}$ is located at the highest row, with its P_4 equal to 5. Hence, the selection order of the next time slot (time slot 8) is $v_{2,2} \rightarrow v_{2,3} \rightarrow v_{3,5}$.

Instead of comparing the priority vectors, for realization, it will be more convenient to map the priority vectors to scalars so that we can compare scalars when prioritizing the associated tasks. Specifically, we map a priority vector $\langle P_1(v_{i,j}), P_2(v_{i,j}), P_3(v_{i,j}), P_4(v_{i,j})\rangle$ to a priority scalar $P(v_{i,j})$ as follows:

$$P(v_{i,j}) = \sum_{h=1}^{H=4} \left(\frac{N}{2}\right)^{H-h} P_h(v_{i,j}). \tag{2.15}$$

Note that for the $H = 4$ MPS, the values of P_1, P_3, P_4 are fixed at the beginning of the scheduling and will never change as the algorithm proceeds. However, P_2 of a vertex $v_{i,j}$ may change dynamically, according to whether the companion vertex of $v_{i,j}$ has already been selected for execution.

Recall that a paired-type vertex shares one or two common children with another companion vertex. If a vertex has children in the initial precedence graph, it has an unselected companion vertex. Hence, there are only two types of vertexes in the initial FFT precedence graph: the childless type and the paired type. With the execution of the MPS algorithm, a paired vertex $v_{i,j}$ will change to a singleton vertex once its companion has been selected for execution. As a result, P_2 of a vertex can be 0 or 1 in the initial FFT precedence graph (depends on whether it has children). Then, with the execution of MPS, the P_2 of a vertex can be adjusted to 2 accordingly.

Formally, the above description of the assignment of priority vectors, and the serialization for the computation of the priority scalar, can be summarized as follows: For $v_{i,j} \in V$, we denote the set of $v_{i,j}$'s children by $K(v_{i,j})$ and the cardinality of $K(v_{i,j})$ by $k_{i,j}$, we then have

1) P_1 assignment: if $k_{i,j} = 0$, then $P_1(v_{i,j}) = 0$; otherwise, we have $P_1(v_{i,j}) = \max_{v_{i+1,x} \in K(v_{i,j})} P_1(v_{i+1,x}) + 1$.

2) P_2 assignment (initial value): For $v_{i,j}$, if $k_{i,j} = 0$, then $P_2(v_{i,j}) = 0$; otherwise, we have $P_2(v_{i,j}) = 1$.

3) P_2 assignment (dynamically adjusted value): If $v_{i,j}$'s companion is newly selected for execution, let $P_2(v_{i,j}) = 2$.

4) P_3 assignment: For $v_{i,j}$, $P_3(v_{i,j}) = k_{i,j}$.

5) P_4 assignment: For $v_{i,j}$, $P_4(v_{i,j}) = \frac{N}{2} - 1 - j$.

6) Priority scalar calculation: $P(v_{i,j}) = \sum\limits_{h=1}^{H=4} \left(\frac{N}{2}\right)^{H-h} P_h(v_{i,j})$.

We denote the set of executable vertexes by Ω and use pseudo-code to describe the $H = 4$ MPS algorithm. See Appendix D for details about pseudo-code implementations, in which Algorithm 1 is the main function of the MPS algorithm, Algorithm 2 is the initialization subfunction called at the beginning of Algorithm 1, Algorithm 3 is the priority scalar calculation subfunction described by (2.15).

2.4 Lower Bound Analysis for IFDMA FFT Computing

We now derive a general lower bound T^L on the IFDMA FFT computation time. Any feasible schedule $\mathbf{X} = (\chi_0, ..., \chi_{T-1})$ constrained by the FFT precedence graph and the number of available processors must have computation time $T \geq T^L$.

A trivial lower bound is,

$$T^L = \left\lceil \sum_{i=0}^{\log N - 1} Q_i \Big/ M \right\rceil , \tag{2.16}$$

where $\lceil \cdot \rceil$ is the round-up operation, and Q_i is the number of vertexes at stage i. This is, however, a rather loose lower bound.

In the following, we prove a tight lower bound for T. Our argument divides the vertexes into "trunk vertexes" and "branch vertexes", defined as follows:

Definition 5 (Trunk Vertex and Branch Vertex)**.** *In an initial FFT precedence graph with β FFT stages*[2]*, if a vertex $v_{i,j}$ satisfies,*

$$P_1(v_{i,j}) = \beta - 1 - i, \tag{2.17}$$

we refer to it as a trunk vertex; otherwise, we refer to it as a branch vertex.

Recall that in Chapter 2.3.2, we define $P_1(v_{i,j})$ to be the number of generations of descendants vertex $v_{i,j}$ has. Since vertex $v_{i,j}$ is located at stage i, and the maximum number of remaining stages after stage i is $\beta-1-i$, in general we have $P_1(v_{i,j}) \leq \beta - 1 - i$. Hence, trunk vertexes at stage i are vertexes at stage i that have the maximum possible P_1.

Fig. 2.6 gives an illustration. Vertexes in blue are the trunk vertexes, and vertexes in red are the branch vertexes. The rationale for our nomenclature is as follows. We picture vertexes in blue to be part of the main trunk of a tree and vertexes in red to be part of branches of a tree.

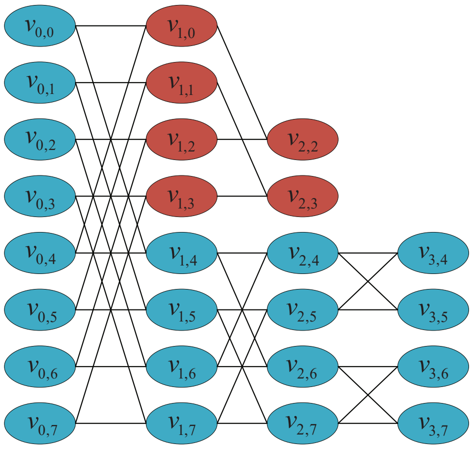


Figure 2.6: An example of "trunk vertexes" and "branch vertexes".

Lemma 1. *A trunk vertex $v_{i,j}$ with $i \in \{0, 1, ...\beta - 2\}$, that is not located at the last stage (i.e., stage $\beta - 1$) has at least one child that is also a trunk vertex.*

---

[2] Since many bin allocations do not require tasks in later FFT stages, we have $\beta \leq \log_2 N$ in general.

Proof. Since $v_{i,j}$ is a trunk vertex, we have

$$P_1(v_{i,j}) = 1 + \max_{v_{i+1,x} \in K(v_{i,j})} P_1(v_{i+1,x}) = \beta - i - 1, \tag{2.18}$$

where $K(v_{i,j})$ is the children set of $v_{i,j}$.

From (2.18), we know that at least one vertex $v_{i+1,x}$ satisfies,

$$P_1(v_{i+1,x})\big|_{v_{i+1,x} \in K(v_{i,j})} = \beta - (i+1) - 1, \tag{2.19}$$

i.e., $v_{i+1,x}$ is a trunk vertex. ■

Let $T_{tr}(i)$, $i = 0, 1, ...\beta - 2$, be the time slot by which all trunk vertexes at stage i are executed.

Lemma 2. $T_{tr}(i+1) \geq T_{tr}(i) + 1$.

Proof. In general, some trunk vertexes at stage i could be executed at time slots earlier than $T_{tr}(i)$. But there must be at least one trunk vertex $v_{i,j}$ at stage i that is executed in time slot $T_{tr}(i)$. From Lemma 1, trunk vertex $v_{i,j}$ must have one child trunk vertex, say vertex $v_{i+1,x}$, at stage $i+1$ that depends on $v_{i,j}$. The earliest time slot by which $v_{i+1,x}$ can be executed is $T_{tr}(i) + 1$. Thus, $T_{tr}(i+1) \geq T_{tr}(i) + 1$. ■

Let R_i be the number of trunk vertexes at stage i. Some stages have more than M trunk vertexes, and some stages have no more than M trunk vertexes. Let $\mathbf{U}$ be the set of stages with more than M trunk vertexes, and $\mathbf{W}$ be the set of stages with no more than M trunk vertexes.

Lemma 3. *In an FFT precedence graph, the stages in* $\mathbf{U}$ *precedes the stages in* $\mathbf{W}$.

Proof. A pair of trunk parent vertexes at stage i have no more than two child trunk vertexes at stage $i+1$. Each child trunk vertexes, on the other hand, must have two parent trunk vertexes. Hence, R_i, the number trunk vertexes at stage i, can be no smaller than R_{i+1}, the number of trunk vertexes at stage $i+1$. Thus, the stages in $\mathbf{U}$ precede the stages in $\mathbf{W}$. ■

We denote the cardinality of the set $\mathbf{W}$ by m in the following. We also refer to the computation time required to execute all trunk vertexes as T_{tr}, and we denote the lower bound of T_{tr} by T_{tr}^L. With the above lemmas, we obtain T_{tr}^L as follows.

Lemma 4. $T_{tr} \geq T_{tr}^{L} \triangleq \left\lceil \sum\limits_{i=0}^{\beta-m-1} R_i \Big/ M \right\rceil + m$.

Proof. From Lemma 3, the first $\beta - m$ stages have more than M in each stage, and their execution can finish no earlier than,

$$T_{tr}\left(\beta - m - 1\right) \geq \left\lceil \sum_{i=0}^{\beta-m-1} R_i \Big/ M \right\rceil . \tag{2.20}$$

Applying Lemma 2 to stage $\beta - m$, we have

$$T_{tr}\left(\beta{-}m\right) \geq T_{tr}\left(\beta{-}m{-}1\right) + 1 \geq \left\lceil \sum_{i=0}^{\beta-m-1} R_i \Big/ M \right\rceil + 1. \tag{2.21}$$

Similarly, for stage $\beta - m + 1$, we have $T_{tr}\left(\beta - m + 1\right) \geq T_{tr}\left(\beta - m\right) + 1$. And so on and so forth. The last trunk vertexes at the stage $\beta - 1$ can therefore finish execution no earlier than $\left\lceil \sum\limits_{i=0}^{\beta-m-1} R_i \Big/ M \right\rceil + m$. Thus, we have

$$T_{tr} \geq T_{tr}^{L} \triangleq \left\lceil \sum_{i=0}^{\beta-m-1} R_i \Big/ M \right\rceil + m. \tag{2.22}$$

■

The T_{tr}^{L} in (2.22) would be a lower bound for the overall problem if there were no branch vertexes. We next take into consideration the computation time needed by the branch vertexes. When packing the vertexes into the schedule implicit in our procedure for obtaining T_{tr}^{L} above, there could be time slots in which there are fewer than M vertexes being executed. As far as a lower bound for the overall computation time is concerned, we could imagine that we could pack branch vertexes into such time slots for execution without constraints of the precedence graph. In other words, we imagine that the branch vertexes could fully exploit the unused capacities of the time slots allocated to the trunk vertexes. The lower bound obtained as such may not be achievable, but it is still valid as far as a lower bound.

The total number of branch vertexes is $\sum\limits_{i=0}^{\beta-1} \left(Q_i - R_i\right)$. If the unused capacities is not enough for executing all branch vertexes, i.e., $M \cdot T_{tr}^{L} - \sum\limits_{i=0}^{\beta-1} R_i < \sum\limits_{i=0}^{\beta-1} \left(Q_i - R_i\right)$, we need extra time slots after T_{tr}^{L} to execute the remaining branch vertexes. We refer to the computation time for executing the remaining branch vertexes as T_{br}, and we denote the lower bound of T_{br} by T_{br}^L.

Lemma 5. *$T_{br} \geq T_{br}^L \triangleq \lceil l/M \rceil$,*
where $l = \max\left\{0, \sum\limits_{i=0}^{\beta-1} Q_i - M\left\lceil \sum\limits_{i=0}^{\beta-m-1} R_i \Big/ M \right\rceil + Mm\right\}$.

Proof. There are $M \cdot T_{tr}^L$ processor capacities in the first T_{tr}^L time slots. During the T_{tr}^L time slots, the unused capacities are given by,

$$E = M \cdot T_{tr}^L - \sum_{i=0}^{\beta-1} R_i. \tag{2.23}$$

After packing branch vertexes into T_{tr}^L time slots to used up the capacities E, the number of remaining branch vertexes not packed into the T_{tr}^L time slots is,

$$\begin{aligned} l &= \max\left\{0, \sum_{i=0}^{\beta-1}(Q_i - R_i) - E\right\} \\ &= \max\left\{0, \sum_{i=0}^{\beta-1} Q_i - M \cdot T_{tr}^L\right\} \\ &= \max\left\{0, \sum_{i=0}^{\beta-1} Q_i - M\left\lceil \sum_{i=0}^{\beta-m-1} R_i/M \right\rceil + Mm\right\} \end{aligned} . \tag{2.24}$$

Thus, T_{br} is lower bounded by,

$$T_{br} \geq T_{br}^L \triangleq \left\lceil \frac{l}{M} \right\rceil. \tag{2.25}$$

■

Theorem 1. *$T \geq T^L \triangleq \left\lceil \sum\limits_{i=0}^{\beta-m-1} R_i \Big/ M \right\rceil + \lceil l/M \rceil + m$.*

Proof. Since $T \geq T_{tr}^L + T_{br}^L$, we prove Theorem 1 with Lemma 4 and Lemma 5. ■

2.5 Experimental Results

In Chapter 2.2.3, we proved that when the number of subcarriers is large, the number of non-isomorphic instances for IFDMA can be huge. It is impractical to test all possible subcarrier allocation instances. Hence, in the following two subsections, we analyze the performance of MPS-FFT statistically when the number of subcarriers is large. Specifically, when $n \geq 7$, we randomly sample $f_6 = 2598061$ of f_n non-isomorphic instances for testing, where f_n is the total number of non-isomorphic instances when there are 2^n subcarriers. For systems of 8 to 64 subcarriers (i.e., $3 \leq n \leq 6$), we test all non-isomorphic instances. The two-point ($n = 1$) and four-point ($n = 2$) FFT modules are trivial and are omitted in the experiments.

2.5.1 Optimality of MPS

We start with the optimality of MPS. We assume a fully packed IFDMA system, in which all subcarriers are used. Recall from Chapter 2.2.3 that $\eta_{n,M}$ denotes the percentage of MPS-FFT schedules that are optimal when the number of subcarriers is 2^n and the number of processors is M. If the computation time does not reach the lower bound, we use $\gamma_{n,M}$ to denote the average percentage gap between the observed computation time and the theoretical lower bound. As we have explained at the beginning of this section, if $n \geq 7$, we follow the analysis in Chapter 2.2.3 and statistically obtain η and γ with a confidence level of $\alpha = 0.95$.

Case 1: arbitrary number of processors: Let us first assume an arbitrary number of processors between one and 2^{n-1}, the maximum processors needed. We denote the average value of $\eta_{n,M}$ and $\gamma_{n,M}$ over M by η_n and γ_n, respectively. Experimental results of η_n and γ_n are given in Table 2.2.

As can be seen from Table 2.2, if we give MPS a random subcarrier allocation instance, MPS can find a schedule reaching the computation-time lower bound with high probability. Specifically, η_n is no less than 98.70%. Additionally, if a task-execution schedule does not reach the lower bound, it is still acceptable because the computation time is very close to the lower bound, i.e., γ_n is no more than 6.48%.

Case 2: power-of-two number of processors: Note that we assume an arbitrary number of processors in Table 2.2. We find from our experiments that for the case of M equal to a power-of-two integer, all our tested task-execution schedules reach the lower bound and are optimal. For $n < 7$, our

Table 2.2: η_n and γ_n for the scheduling problems in a 2^n-point IFDMA FFT (the case of an arbitrary number of processors)

n	η_n	γ_n	n	η_n	γ_n
3	1.0000	0.0000	7	≥ 0.9972	≤ 0.0391
4	0.9985	0.0250	8	≥ 0.9942	≤ 0.0146
5	0.9984	0.0460	9	≥ 0.9901	≤ 0.0121
6	0.9981	0.0648	10	≥ 0.9870	≤ 0.0104

algorithm is optimal in terms of the computation speed, as we tested all instances, and all the schedules reached the lower bound. For $n \geq 7$, since $\xi_0 - \xi_1 = 0$, (2.11) in Chapter 2.2.3 can be further simplified as follows:

$$\begin{aligned} I_{\eta_n^L}\left(\xi_1+1, \xi_0-\xi_1+1\right) &= \sum_{\substack{j=\xi_1+1\\ \xi_0=\xi_1}} \frac{(\xi_0+1)!}{j!\,(\xi_0+1-j)!} \left(\eta_n^L\right)^j \left(1-\eta_n^L\right)^{\xi_0+1-j} \\ &= \left(\eta_n^L\right)^{\xi_0+1} \end{aligned} \tag{2.26}$$

Applying $\xi_0 - \xi_1 = 0$ to (2.8) and (2.11), we have

$$\eta_n^L = (1-\alpha)^{\frac{1}{\xi_0+1}}. \tag{2.27}$$

Therefore, when $n \geq 7$, we have $\eta_n \geq (1-\alpha)^{\frac{1}{\xi_0+1}}$, where $\alpha = 0.95$ is the confidence level, and $\xi_0 = f_6 = 2598061$. Additionally, we know that $\gamma_n = 0$ from the definition of $\gamma_{n,M}$ in (2.2). The experimental results of η_n and γ_n are given in Table 2.3.

The gap between $\sqrt[\xi_0+1]{0.05}$ and one is relatively small because ξ_0 is large enough, i.e., the lower bound of η_n is relatively close to one. Thus, we can say that our algorithm has close-to-optimal performance when the number of processors is power-of-two.

2.5.2 Applying MPS-FFT in Compact IFDMA Transceivers

We now analyze the effect of applying MPS-FFT to compact IFDMA transceivers. To distinguish the transceiver design in this paper from the design in our

Table 2.3: η_n and γ_n for the scheduling problems in a 2^n-point IFDMA FFT (the case of a power-of-two number of processors)

n	η_n	γ_n
3/4/5/6	1.0000	0.0000
7/8/9/10	$\geq \sqrt[\xi_0+1]{0.05}$	0.0000

previous paper [1], let us refer to the compact IFDMA transceiver in our previous work and the one in the current paper as compact IFDMA transceiver v1 and v2, respectively.

As has been explained at the beginning of this section, for $n \geq 7$, we analyze the computation complexity/hardware utilization/computation time of MPS-FFT with a subset of f_6 randomly sampled non-isomorphic subcarrier allocation instances rather than the full set of instances. We show in Appendix C that our subset generation policy selects all the instances with equal probability. Hence, the analysis of computation complexity/hardware utilization/computation time is unbiased in terms of instance selection.

Computation complexity: The computation complexity improvement of compact IFDMA transceiver v1 over conventional IFDMA transceivers has already been studied in Section IV and Section V of our previous work [1]. For readers' convenience, we briefly summarize the results in Table 2.4. In the table, (i), (ii), and (iii) refer to conventional IFDMA transceivers with time-domain realization, conventional IFDMA transceivers with frequency-domain realization, and compact IFDMA transceivers v1, respectively. The computation complexity is measured in terms of the number of butterfly computations needed in a transmitter/receiver. As can be seen from Table 2.4, transceiver v1 reduces the complexity of a conventional IFDMA transceiver by at least a factor of $\log_2 N$.

The current paper further reduces complexity by pruning the FFT flow graph. Fig. 2.7 below compares the computation complexities in compact IFDMA transceiver v2 and IFDMA transceiver v1. As the figure shows, transceiver v2 achieves at least 11.21% reduction in FFT computation complexity over transceiver v1.

Table 2.4: Complexities of Conventional IFDMA Transceivers and Compact IFDMA Transceiver v1

	(i)	(ii)	(iii)
Receiver	$\frac{1}{4}N\log_2^2 N + \frac{1}{2}N\log_2 N$	$\frac{1}{4}N\log_2^2 N + \frac{1}{2}N\log_2 N$	$\frac{1}{2}N\log_2 N$
Transmitter	N^2	$\frac{1}{4}N\log_2^2 N + \frac{1}{2}N\log_2 N$	$\frac{1}{2}N\log_2 N$

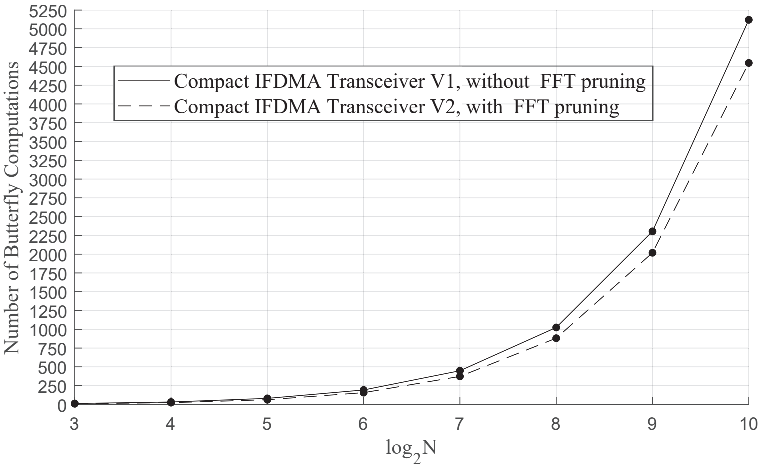


Figure 2.7: Computation complexity of compact IFDMA transceiver v1 and compact IFDMA transceiver v2.

Hardware Utilization and Computation Time: Although pruning the FFT flow graph reduces the computation complexity, it was only the first step. A more challenging study of compact IFDMA transceiver v2 is to increase the hardware utilization in a pruned FFT with a limited number of processors.

As we have explained, the goal of our scheduling algorithm is to produce a schedule with as short a computation time as possible for any given IFDMA subcarrier allocation. Since the number of hardware processors is limited, MPS achieves that goal by boosting the hardware utilization of the processors during the FFT processing. In the FFT precedence graph, one butterfly computation cannot be executed unless its two parents have been executed. In some rounds of butterfly computation during the FFT processing, it is possible that some processors have no work to do if the schedule is not well constructed. This is because there may not be enough executable computations for every processor to get a job in this round if the schedule is poorly constructed. The “idle time” of processors reduces the hardware utilization. Additionally, conventional FFT may execute unnecessary butterfly computations (i.e., those eliminated in our pruned FFT).

In essence, we define hardware utilization as $\sum Q_i/MT$, where Q_i is the number of necessary computation tasks in FFT stage i, T is the overall computation time required by the schedule, and MT is the total number of available hardware capacity over that time.

Fig. 2.8 benchmarks compact IFDMA transceiver v1 and v2 in terms of hardware utilization. The FFT module used in transceiver v2 is MPS-FFT designed in this paper. As for the FFT used in compact IFDMA transceiver v1, our prior work [1] investigated the computation complexity but did not consider the implementation issue. It just assumed a conventional FFT but did not assign a specific realization scheme. Basically, there are three types of conventional FFT implementations: the serial structure [27], the pipelined structure [28], and the parallel structure [29]. Among the three types, the pipelined FFT is the most widely used in practice, e.g., the same realization can be found in Xilinx’s high-efficiency FFT IP core for FPGA [46]. As for the other two types of FFT realizations, serial FFTs are too slow for the FFT processing in communication systems; parallel FFTs consume too much hardware resource and are not suitable for communication systems with many subcarriers.

Here IFDMA transceiver v1 adopts the pipelined FFT implementation, while IFDMA transceiver v2 adopts this paper’s MPS-FFT.

In Fig. 2.8, we consider IFDMA transceivers with N subcarriers and plot the figure with $\log_2 N$ on the x-axis. Note that the number of processors in an N-point conventional pipelined FFT is fixed to be $\log_2 N$. For a fair comparison, we let the number of hardware processors in both transceiver v1 and transceiver v2 be $M = \log_2 N$.

As can be seen from the dashed line in Fig. 2.8, the hardware utiliza-

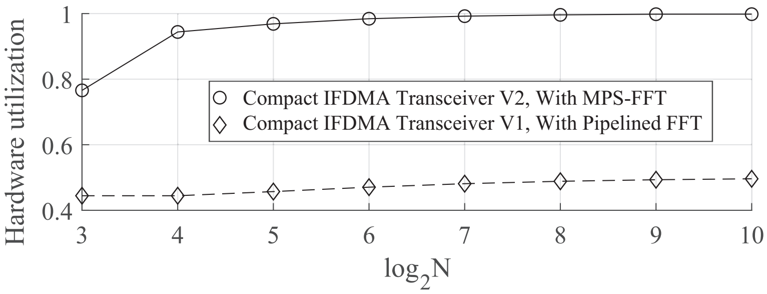


Figure 2.8: Experimental results showing that MPS-FFT outperforms pipelined FFT in terms of hardware utilization.

tion of a conventional pipelined FFT is no larger than 50%. MPS-FFT, on the other hand, greatly increases the hardware utilization. Specifically, the hardware utilization rates of MPS-FFT with 64 or more points are at least 98.42%. And the utilization rate increases as number of subcarriers increases, reaching 99.82% in the 1024-point FFT case.

Table 2.5 compares compact IFDMA transceiverz v1 and v2 in terms of computation time. We can see that compact IFDMA transceiver v2 achieves much lower FFT computation time than compact IFDMA transceiver v1 does. Specifically, the computation time can be reduced by at least 55.87%, and we can calculate that the average computation time reduction rate is 59.85%.

2.6 Summary

This chapter studied the FFT relization of an efficient IFDMA transceiver, referred to as compact IFDMA, put forth by a recent investigation. For compact IFDMA, not all butterfly computations inside the full FFT network need to be executed, and the necessary computations vary with different subcarrier allocations. When applied to IFDMA, conventional FFT implementation is resource-wasteful because it does not exploit this specific property of IFDMA signal processing.

This chapter focused on FFT implementations tailored for IFDMA. We put forth a flexible heuristic algorithm to schedule the butterfly computations in IFDMA FFT, referred to as multi-priority scheduling (MPS).

Table 2.5: Number of Average Time Slots Needed for the FFT computation in Compact IFDMA Transceivers v1/v2

n	FFT computation time in transceivers v1	FFT computation time in transceivers v2	Reduction rate
3	9	3.70	0.5889
4	18	6.12	0.6600
5	35	12.84	0.6331
6	68	26.48	0.6106
7	133	54.25	0.5921
8	262	110.61	0.5778
9	519	224.76	0.5669
10	1032	455.42	0.5587

Compared with conventional FFT implementations, the FFT computation schedule obtained by MPS, referred to as MPS-FFT, has two advantages: 1) MPS-FFT reduces computation requirements. For example, for 1024-subcarrier IFDMA, MPS-FFT can bypass at least 11.21% of the computation tasks in FFT; 2) MPS-FFT utilizes hardware efficiently. For example, for 1024-subcarrier IFDMA, the processor utilization rate in MPS-FFT is 99.82%, much higher than the processor utilization rate in conventional pipeline FFT.

When the number of available processors is a power of two, MPS-FFT has near-optimal computation-time performance. Quantitatively, MPS-FFT subject to a randomly selected IFDMA subcarrier allocation can reach the computation time lower bound with a probability larger than $\sqrt[\xi_0+1]{0.05}$, where ξ_0 is more than 2.5 million in our experiment. Furthermore, MPS-FFT incurs less than 44.13% of the computation time of the conventional pipelined FFT.

Chapter 3

Reliable Packet Detection in CSMA

3.1 Background

Wireless random access technology offers significant advantages for vehicular applications [9, 47, 48]. It facilitates efficient communications among a vast array of devices, from intelligent vehicles to roadside sensors. Unlike centralized access control, random access enables massive machine-type communications (MTC) intrinsic to smart transportation networks without the need for a predetermined transmission schedule. For instance, in traffic monitoring systems, a sensor embedded within the infrastructure or onboard a vehicle may generate a new packet only upon detecting an anomaly and then transmit this information to a central monitoring system through the wireless channel. As the transmission is sporadic and unpredictable, employing random access is more efficient than pre-allocating dedicated wireless resources (e.g., time slots or subcarriers) to each sensor. Moreover, centralized access is impractical for scheduling every sensor when the number of connections exceeds the available wireless resources.

In random access, a receiver does not know when a wireless device will transmit a packet to it. To conserve power and avoid being wrongly occupied, the packet decoding circuitry of a receiver should not get activated unless a packet is being transmitted, i.e., the receiver needs to detect the incoming packet before decoding it. Therefore, there are two primary causes for reception failures in random access: 1 the payload of the packet cannot be decoded due to noises or transmission errors; or 2) the payload of the packet can be decoded with advanced decoding schemes, but the packet is not even detected.

A recent trend in the vehicular domain is to use packets with short payloads to reduce decoding error probability [49, 50, 10]. These research efforts significantly overcome the first type of failure but ignore possible reception failure in the packet detection process. This paper is an attempt to address the second type of failure and enhance the overall reliability in vehicular random-access networks. We revisit the conventional Schmidl-and-Cox (S&C) algorithm [11] widely applied as the underlying scheme for many recent packet-detection studies [12, 13, 14, 15, 16, 17, 18, 19, 20, 21]. We point out the limitations of this conventional algorithm and related prior works on packet detection and put forth our solutions in the following:

The first limitation is the lack of a rigorous analysis framework, which further results in the absence of closed-form expressions for missed-detection and false-alarm probabilities. Previous research [12, 13, 14, 15, 16, 17, 18, 19, 20, 21] used the ratio of autocorrelation and signal power as the packet-detection metric. When the metric is above a preset threshold, the packet-detection process at the receiver then declares that there is an incoming packet. The value of the threshold determines the trade-off between the false-alarm and missed detection probabilities. Unfortunately, the ratio contains two random variables whose noise terms are correlated, making the ratio metric difficult to analyze, and the probabilistic analysis of false alarms and missed detections becomes intractable as a result. When the distribution of this metric is unknown, a rigorous analysis of the packet detection, particularly the trade-off between the probabilities of false alarm and missed detection, is not possible.

This paper proposes a new “sum metric” for packet detection called the “compensated autocorrelation”. The new metric is equivalent to the previous ratio metric as far as packet detection probability is concerned. However, the noise characteristic of the compensated autocorrelation is analytically tractable, as the metric contains only a simple summation of correlated noises, as opposed to the convoluted relationship of the correlated noises in the numerator and the denominator of the ratio metric. Importantly, our sum metric can be approximated as a Gaussian random variable. Hence, the introduction of the new metric makes possible a rigorous analytical framework not previously reported in the literature. We demonstrate through experiments that our derivations and approximations are precise and accurate. Importantly, the expression of our metric suggests a way to extend single-antenna packet detection to the multi-antenna packet detection setting (details presented in the "fourth limitation" below).

The second limitation is a deficiency in prior works' benchmark method for packet detection algorithms. Previous research focused on minimizing the missed-detection probability as the sole criterion. For example, in [51], the authors conclude that one algorithm is better than the other due to its low missed-detection probability. However, this conclusion may be misleading because low missed-detection probability may come at the expense of high false-alarm probability[1], and one could in principle set the packet-detection threshold to a very small value to lower the missed-detection probability arbitrarily if false-alarm probability were not a concern.

Inspired by the concept of Pareto optimality [52], we address the limitation with a new benchmarking scheme. Our benchmark scheme plots the false-alarm probability and the missed-detection probability as the x-axis and the y-axis of a figure, allowing for simultaneous consideration of false alarms and missed detections. With the new scheme, we benchmark two packet-detection algorithms in a rigorous and fair manner. Further, we introduce a method to find the optimal packet-detection threshold for our packet-detection metric consistent with our benchmark method (see Appendix H for details).

The third limitation is the use of the magnitude when calculating the

---

[1] Avoiding false alarms is as important as preventing missed detections. First, when a false alarm occurs, signal processing circuits are erroneously activated, leading to a decrease in power efficiency. Second, to avoid packet collisions, a random-access device may hold back and refrain from transmitting a packet itself upon encountering a false alarm, resulting in reduced spectrum efficiency. Third, during the false alarm period, as the receiver is occupied decoding the "fake packet", all true incoming packets will not get processed until the receiver realizes the situation and resets its state machine.

signal autocorrelation term required in the packet-detection metric. In the absence of carrier frequency offset (CFO) between the transmitter and the receiver, the autocorrelation term has no imaginary part. But in the presence of CFO, part of the autocorrelation term shifts from the real part to the imaginary part (see Appendix F for details), lowering the probability of packet detection if only the real part of the autocorrelation were considered. At the time the S&C algorithm was first developed, hardware devices had very large CFOs due to the limitation of semiconductor technologies [53, 54]. Hence, the S&C algorithm considers the magnitude of the autocorrelation to include the CFO-induced autocorrelation leak to the imaginary part.

Taking the magnitude of the autocorrelation is beneficial when the CFO is relatively large, as it reduces potential missed packet detections. However, if we also consider false alarms in the benchmark, such a benefit comes at the price of a higher false-alarm probability. That is, when there is no packet over the air, taking the magnitude of the autocorrelation naturally introduces an additional 3dB noise than taking the real part of the autocorrelation only, increasing the false-alarm probability. With the advancements in hardware and semiconductor technologies, communication devices nowadays have significantly lower CFOs than before [55, 56, 57]. We propose to exploit the improved CFO performance by taking only the real part of the autocorrelation in the packet detection process to strike for a better missed-detection-false-alarm trade-off. We show through experiments that the proposed scheme, referred to as the real-part S&C (RP-S&C) algorithm, exhibits superior performance to the conventional S&C under realistic CFO settings of modern hardware.[2]

The fourth limitation involves the optimization of packet detection algorithms in the multi-antenna scenario. The conventional S&C algorithm was originally proposed for single-antenna receivers. In modern communication systems, receivers are typically equipped with multiple antennas,

---

[2] While RP-S&C is motivated by the improved technology with less CFO, another side benefit is related to the metric of "compensated autocorrelation", which allows a cleaner expression of the compensated autocorrelation without the complicated non-linear complex-number magnitude, thus enabling detailed noise analysis and extension to the multi-antenna setting (see footnote five).

enabling the possibility of enhancing system reliability via rich spatial diversity. While previous multi-antenna investigations [58, 59, 60] have delivered higher decoding reliability by leveraging spatial diversity, packet detection in multi-antenna systems has received less attention. Some previous studies tried to extend the S&C algorithm to the multi-antenna scenario [15-18]. However, lacking a rigorous analytical framework for the packet-detection process, these works typically combine the autocorrelations obtained at different antennas in an intuitive but *ad hoc* manner without a rigorous optimization analysis. Further, these works focused solely on the packet-detection probability in performance evaluation by simulation, without considering the false-alarm probability caused by noise.

We extend our framework built upon the compensated autocorrelation metric to advanced multi-antenna systems. The aggregated noise in the weighted sum of the individual compensated autocorrelations obtained at different antennas still contains a sum of noises only, and is therefore analytically tractable (note: this is not the case if RP-S&C is not used and the magnitude of autocorrelation is used in the compensated autocorrelation expression). Using the weighted sum as the metric in the multi-antenna scenario is a natural extension of the single-antenna treatment, and optimality under different criteria can be established rigorously. This paper considers two specific criteria: 1) minimizing false-alarm probability and 2) minimizing missed-detection probability. We then give our solutions, the weight assignment for minimizing false alarms (WFA) and the weight assignment for minimizing missed detections (WMD), to the two optimization problems with rigorous proofs. Apart from concrete theoretical analyses, we also conduct extensive emulation experiments with real-world channel data to benchmark the two schemes. We therefore recommend the WFA as the preferable choice due to its reliability, simplicity and superior packet-detection performance.

3.2 RP-S&C for Single-antenna Packet Detection

3.2.1 Conventional S&C Algorithm

A random-access system employs repeating sequences to detect packets. Fig. 3.1 shows a general packet format for random access. The repeating sequences at the beginning of a packet are referred to as short training sequences (STSs), and a collection of multiple STSs forms the preamble sequence. Let us denote the number of STSs by m and the length of each STS by η. In this chapter, for simplicity, we assume that the preamble sequence contains two STSs, i.e., $m = 2$. There are several ways to extend the basic treatment here to more general preamble sequences with more than two STSs.

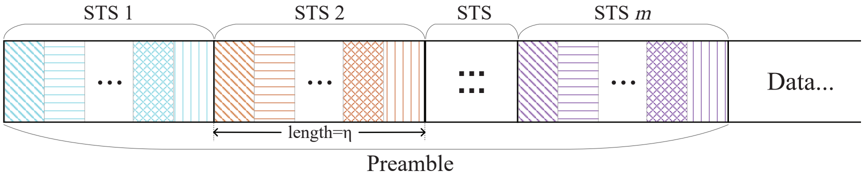


Figure 3.1: A general packet format for random access.

The conventional S&C algorithm uses a packet-detection metric $l[n]$ written as

$$l[n] = |a[n]|/b[n] \tag{3.1}$$

where $a[n]$ and $b[n]$ denote the autocorrelation and the average signal power over the two STSs, respectively. By definition, we have $a[n]$ and $b[n]$ as

$$a[n] = \frac{1}{\eta}\sum_{k=0}^{\eta-1} y[n+k]{\cdot}y^*[n+\eta+k] \tag{3.2}$$

$$b[n] = \frac{1}{2\eta}\sum_{k=0}^{2\eta-1} y[n+k]{\cdot}y^*[n+k] \tag{3.3}$$

where $y[n]$ denotes samples captured by the wireless receiver.

In ideal cases (i.e., without noise or CFO), as the transmitted preamble sequence repeats itself every η samples, $l[n]$ reaches its peak value of one at the particular index n corresponding to the beginning of the first STS sample. For non-ideal cases, on the other hand, $l[n]$ is in general smaller than one (to be elaborated later in this chapter). The conventional S&C algorithm compares $l[n]$ with a pre-defined threshold ρ. If $l[n] > \rho$, S&C declares the detection of a packet; otherwise there is no packet. Fig. 3.2 and Fig. 3.3 illustrate the packet-detection process.

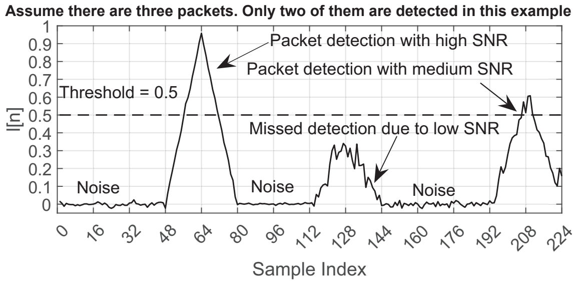


Figure 3.2: Illustration of how S&C detects wireless packets with an example of missed packet detection. Here $\eta = 16$.

In Fig. 3.2, we assume that there are three packets. For an incoming packet, if the peak of its $l[n]$ is larger than threshold ρ (e.g., the first and the last $l[n]$ peak in Fig. 3.2), then the receiver declares a packet is detected, which triggers the subsequent signal processing to decode the packet. Otherwise, just like the second $l[n]$ peak in Fig. 3.2, the receiver performs no action, and an event of missed packet detection occurs. Packet missed detections may occur often when the antenna signal-to-noise ratio (SNR) is too low or ρ is too high.

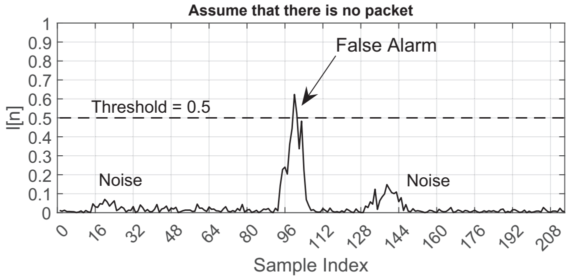


Figure 3.3: Illustration of how false alarms arise.

In Fig. 3.3, we assume there is no packet (i.e., pure noise input). We see that $l[n]$ is very close to zero because the input noise is random. Nevertheless, we may still observe a $l[n]$ larger than ρ, and when that occurs, a false alarm event occurs. False alarms are more likely to occur the lower the threshold ρ. Hence, adjusting the value of ρ amounts to trading off missed-detection performance and false-alarm performance.

3.2.2 Our Improvement over the Conventional S&C Algorithm

For the conventional S&C algorithm given in (3.1), $|a[n]|$ is considered in the calculation of $l[n]$. This chapter first explains why taking the absolute value of $a[n]$ is beneficial at the time the conventional S&C algorithm was first developed. We will then explain why doing so is no more attractive with recent years' improved technology in oscillators. Finally, we explain how our method can have improvements over the conventional scheme by considering only the real part of $a[n]$.

Let us begin by constructing the mathematical model of the received signal $y[n]$. We assume that the two-STS preamble sequence $s[n]$ is normalized, i.e.,

$$\overline{|s[n]|^2} = \frac{1}{\eta}\sum_{n=0}^{\eta-1} |s[n]|^2 = 1 \tag{3.4}$$

At the transmitter side, we denote the power of the transmitted signal, which is amplified by the power amplifier (PA) and the transmitter antenna [61], by P_{TX}, and the power gain of the wireless channel by g_{ch}. We denote the power gain introduced by the receiver antenna and the low-noise amplifier (LNA) at the receiver by g_{RX}. We write the received preamble as[3]

$$y[n] = \sqrt{P}s[n]e^{j\theta[n]} + w[n] \tag{3.5}$$

where $w[n] \sim CN(0, \sigma^2)$ is the complex receiver noise, $e^{j\theta[n]}$ denotes sample $s[n]$'s phase shift caused by CFO, and P is the power of the received signal written as

$$P = P_{TX} \cdot g_{ch} \cdot g_{RX} \tag{3.6}$$

---

[3] A practical slow fading channel may introduce a constant phase offset to all samples within a packet. While this offset may vary between packets, it remains constant throughout any given packet. As explained in Appendix H, this has no impact on packet detection metric and performance.

The phase shift term $e^{j\theta[n]}$ in (3.5) captures the phases introduced by the wireless channel, the transmitter and receiver circuitries, and the CFO. However, if the phases introduced by the wireless channel and the transmitter-receiver circuitries are constant throughout the STS sequence (i.e., the practical slow-fading scenario), then the autocorrelation as computed by (3.2) will only contain a phase term that depends only on the time-varying part of $\theta[n]$ only (the component of $\theta[n]$ that varies with n), and this is due to CFO. In other words, with CFO, the autocorrelation signal as computed by (3.2) can spread into an imaginary part. In Appendix F, we present the analysis on how the rotation of $y[n]$ shifts part of $a[n]$'s power to the imaginary part. To mitigate the impact of CFO, the conventional S&C algorithm uses the absolute value of $a[n]$ to account for the power in its imaginary part.

From a hardware perspective, CFO is introduced by the frequency discrepancy between the crystal oscillators (XOs) used in the transmitter side and the receiver side [62].[4] At the time the classic S&C algorithm was developed (i.e., 1997), semiconductor and hardware technologies for XO production were less advanced than nowadays. For example, in the IEEE 802.11a standard released in 1999, the maximum XO inconsistency allowed is 20 parts per million (ppm) [65]. Given an XO with 20 ppm accuracy, considering the imaginary part of $a[n]$ is more advantageous for packet detection because a significant portion of the signal power may spread into the imaginary part of $a[n]$ due to the large CFO.

However, with significant advancements in hardware and semiconductor technologies during the past decades, modern communication systems have a much lower CFO than the old systems developed in the 1990s [55, 56, 57]. When CFO is relatively small, $\theta[n]$ does not vary much with n, and $a[n]$ is contained mainly in its real part. In such cases, taking the real part of $a[n]$ does not harm the probability of a packet being detected, as the imaginary part of $a[n]$ contains predominantly the noise term only. Moreover, when

---

[4] Even if the two XOs have the same rated value and come from the same product line, they may differ slightly due to manufacturing variances and differences in working conditions, such as temperature [63]. The inconsistency cannot be completely removed, but it can be reduced with the improvement of semiconductor and hardware technologies [64].

no packet is being transmitted, disregarding the imaginary part will reduce the detected noise power by an average of 50% (given that the phase of a noise vector is random), thus making the receiver less prone to false alarms.

As we will later show through experiments, under a practical CFO condition of a modern communication system, taking the real part of $a[n]$ is more beneficial than taking the absolute value of $a[n]$. This is true even when CFO cannot be fully eliminated as in the ideal case (see Chapter 3.3 for details). That is, as long as the XO's inconsistency is within a practical constraint, the gain of ignoring the imaginary part (i.e., significantly lower false-alarm probability) outweighs the loss (i.e., slightly higher missed-detection probability), thus making our scheme more competitive than the conventional S&C algorithm in terms of the packet-detection performance evaluated by the Pareto comparison.

Assuming CFO is negligible, then $\theta[n]$ can be treated as constant throughout n. In particular, a constant $e^{j\theta}$ would not have an effect on $a[n]$ in (3.2), as we have $e^{j\theta} \cdot e^{-j\theta} = 1$. Given this, we can simply remove the term $e^{j\theta}$ in (3.5) and write it as

$$y[n] = \sqrt{P}s[n] + w[n] \tag{3.7}$$

Given that the term $e^{j\theta}$ would not have an effect on the computation of $a[n]$ and $b[n]$ in (23.2) and (3.3), we have

$$l_R[n] = a_R[n]/b[n] \tag{3.8}$$

where the subscript R represents the real part of a variable. We refer to the modification we made over the conventional S&C algorithm as the real-part S&C (RP-S&C) algorithm.

In our later experiments presented in Fig. 3.7, we will replace the zero-CFO assumption with practical CFO conditions that represent real-world scenarios. Experimental results therein prove that RP-S&C has superior performance than conventional S&C even in the presence of CFOs, as long as the CFO value is within the tolerance of modern technologies.

For packet detection, we are interested in whether $l_R[n] > \rho$. Yet, analyzing $l_R[n]$ (also, the original $l[n]$) is challenging as it is the ratio of two non-independent random variables. We note that saying $l_R[n] > \rho$ is equivalent to saying

$$r[n] \triangleq a_R[n] - \rho b[n] \ > 0 \tag{3.9}$$

We refer to $r[n]$ as the "compensated autocorrelation".[5] That is, $a_R[n]$ is the real part of the autocorrelation and we compensate it by subtracting $\rho b[n]$ from it. Having a large $a_R[n]$ does not necessarily mean that there is a packet because it could be due to a large exogenous interference (e.g., Bluetooth interference on WiFi). However, large exogenous interference also has relatively larger $b[n]$ compared with $a_R[n]$, and hence $r[n]$ is likely to be small in that case.

There are two key advantages of focusing on $r[n]$ in (3.9) rather than $l_R[n]$ in (3.8) (or $l[n]$ in (3.1), which is even more complex analytically). First, compared with $l_R[n]$, $r[n]$ is much easier to analyze. The reason $l_R[n]$ is difficult to analyze is its ratio form. Meanwhile, $r[n]$ is a simple linear combination of $a_R[n]$ and $b[n]$, and it can be approximated as a Gaussian random variable (elaborated below). Second, for a multi-antenna system, we could add the weighted $r[n]$ of different antennas to form a weighted-combined $r[n]$ and compare that with a threshold for packet detection purposes. Again, the weighted-combined $r[n]$ is amenable to rigorous noise analysis since the weighted combination can also be approximated as a Gaussian random variable. This allows us to investigate the optimality of different weight combinations on a rigorous basis.

3.2.3 Analysis of RP-S&C with Gaussian Approximation

We now analyze $b[n]$, $a_R[n]$, and their cross term $a_R[n]b[n]$. Based on these analyses, we obtain the mean and variance of $r[n]$. We then approximate $r[n]$ as a Gaussian random variable with the computed mean and variance. Finally, we utilize the distribution of $r[n]$ to determine the false-alarm probability and the missed-detection probability.

---

[5] Note that if we were to define the compensated autocorrelation as $r[n] = |a[n]| - \rho b[n]$ to correspond to the original S&C, $|a[n]|$ would consist of a non-linear expression of its real and imaginary parts, complicating the noise analysis and extension to the multi-antenna case later. With (3.9), though the noise analysis is still hard, it is at least tractable, as will be shown later.

Let us start with $b[n]$. We have that

$$E\left(b[n]\right) = P + \sigma^2 \tag{3.10}$$

For the analysis of $Var\left(b[n]\right)$, we write $b[n]$ as

$$\begin{aligned}
b[n] &= \frac{1}{2\eta}\sum_{k=0}^{2\eta-1} y[n+k]\cdot y^*[n+k] \\
&= \frac{1}{2\eta}\sum_{k=0}^{2\eta-1}\left(\begin{array}{l} P|s[n+k]|^2 + \sqrt{P}s[n+k]w^*[n+k] \\ \quad + \sqrt{P}s^*[n+k]w[n+k] + w[n+k]w^*[n+k] \end{array}\right) \\
&= P + \frac{\sqrt{P}}{\eta}\sum_{k=0}^{2\eta-1}\left(s_R[n+k]w_R[n+k] + s_I[n+k]w_I[n+k]\right) \\
&\qquad + \frac{1}{2\eta}\sum_{k=0}^{2\eta-1}\left(w_R^2[n+k] + w_I^2[n+k]\right)
\end{aligned} \tag{3.11}$$

where subscript R and subscript I represent the real and imaginary parts of a term, respectively.

Considering that $E\left(w_R^3[n]\right) = E\left(w_I^3[n]\right) = 0$ and that $w_I(n)$ and $w_R(n)$ are i.i.d, we write $E\left(b^2[n]\right)$ as in (3.12).

$$\begin{aligned}
&E\left(b^2[n]\right) \\
&= E\left\{\left(P + \frac{\sqrt{P}}{\eta}\sum_{k=0}^{2\eta-1}\left(\begin{array}{c} s_R[n+k]w_R[n+k] \\ +s_I[n+k]w_I[n+k] \end{array}\right) + \frac{1}{2\eta}\sum_{k=0}^{2\eta-1}\left(\begin{array}{c} w_R^2[n+k] \\ +\, w_I^2[n+k] \end{array}\right)\right)^2\right\} \\
&= \left(P^2 + 2P\sigma^2 + \frac{P}{\eta}\sigma^2\right) + \frac{E\left(w_R^4[n]\right) + E\left(w_I^4[n]\right) + 2E\left(w_R^2[n]\right)E\left(w_I^2[n]\right)}{2\eta} \\
&\qquad + \frac{4(2\eta-1)E\left(w_R^2[n]\right)\cdot E\left(w_I^2[n]\right)}{2\eta} \\
&= \left(P^2 + \frac{2\eta+1}{\eta}P\sigma^2\right) + \frac{E\left(w_R^4[n]\right)}{\eta} + \frac{E^2\left(w_R^2[n]\right)}{\eta} + \frac{2(2\eta-1)E^2\left(w_R^2[n]\right)}{\eta} \\
&= P^2 + \frac{2\eta+1}{\eta}P\sigma^2 + \frac{E\left(w_R^4[n]\right)}{\eta} + \frac{\sigma^4/4 + 2(2\eta-1)\sigma^4/4}{\eta}
\end{aligned} \tag{3.12}$$

Note that random variable $\sqrt{2/\sigma^2}w_R$ is of the standard normal distribution. Thus, $(2/\sigma^2)w_R^2[n]$ is a chi-square distribution of degree 1, and we have

$$\begin{aligned}
E\left[\left(\frac{2}{\sigma^2}w_R^2[n]\right)^2\right] &= Var\left(\frac{2}{\sigma^2}w_R^2[n]\right) + E^2\left(\frac{2}{\sigma^2}w_R^2[n]\right) = 3 \\
&\Rightarrow E\left(w_R^4[n]\right) = 3\sigma^4/4
\end{aligned} \tag{3.13}$$

Substituting (3.13) into (3.12), we have

$$E\left(b^2[n]\right) = \left(P^2+2P\sigma^2+\sigma^4\right) + \frac{2P\sigma^2+\sigma^4}{2\eta} \tag{3.14}$$

With (3.14), we have

$$Var\left(b[n]\right) = E\left(b^2[n]\right) \; - E^2\left(b[n]\right) \; = \frac{2P\sigma^2+\sigma^4}{2\eta} \tag{3.15}$$

We next look at $a_R[n]$. We can write $a[n]$ as

$$\begin{aligned} a[n] &= \tfrac{1}{\eta}\sum_{k=0}^{\eta-1} y[n-k]\cdot y^*[n-\eta-k] \\ &= P + \tfrac{1}{\eta}\sum_{k=0}^{\eta-1}\left\{ \begin{array}{l} \sqrt{P}\left(s[n-k]w^*[n-\eta-k] + s^*[n-k]w[n-k]\right) \\ \quad + w[n-k]w^*[n-\eta-k] \end{array} \right\} \end{aligned} \tag{3.16}$$

We extract the real part of (3.16) and write $a_R[n]$ as

$$a_R[n]{=}P{+}\tfrac{1}{\eta}\sum_{k=0}^{\eta-1}\left\{ \begin{array}{c} \sqrt{P}\left(\begin{array}{c} s_R[n{-}k]w_R[n{-}\eta{-}k] \\ +s_I[n{-}k]w_I[n{-}\eta{-}k] \end{array} \right) \\ +\sqrt{P}\left(\begin{array}{c} s_R[n{-}k]w_R[n{-}k] \\ +s_I[n{-}k]w_I[n{-}k] \end{array} \right) \\ +\, w_R[n{-}k]w_R[n{-}\eta{-}k] \\ +\, w_I[n{-}k]w_I[n{-}\eta{-}k] \end{array} \right\} \tag{3.17}$$

From (3.17), as zero-mean Gaussian noise terms at different time indexes are independent, we have

$$E\left(a_R[n]\right) = P \tag{3.18}$$

Similarly, we have

$$\begin{aligned} E\left(a_R^2[n]\right) &{=}P^2{+}\frac{1}{\eta^2}\sum_{k=0}^{\eta-1}\left\{ \begin{array}{c} PE\left(\begin{array}{c} \left(\begin{array}{c} s_R^2[n{-}k]w_R^2[n{-}\eta{-}k] \\ +s_I^2[n{-}k]w_I^2[n{-}\eta{-}k] \end{array} \right) \\ +\left(\begin{array}{c} s_R^2[n{-}k]w_R^2[n{-}k] \\ +s_I^2[n{-}k]w_I^2[n{-}k] \end{array} \right) \end{array} \right) \\ +E\left(\begin{array}{c} w_R^2[n{-}k]w_R^2[n{-}\eta{-}k] \\ +w_I^2[n{-}k]w_I^2[n{-}\eta{-}k] \end{array} \right) \end{array} \right\} \\ &{=}P^2{+}\frac{P\sigma^2}{\eta}{+}\frac{\sigma^4}{2\eta} \end{aligned} \tag{3.19}$$

From (3.18) and (3.19), we get

$$Var(a_R[n]) = E(a_R^2[n]) - E^2(a_R[n]) = \frac{2P\sigma^2 + \sigma^4}{2\eta} \tag{3.20}$$

We next look at $a_R[n]b[n]$. With (3.11) and (3.17), we obtain $E\left(a_R[n] \cdot b[n]\right)$ as in (3.21).

$$\begin{aligned}
&E\left(a_R[n] \cdot b[n]\right) \\
&= E\left\{ \begin{array}{l} \left(P + \frac{\sqrt{P}}{\eta} \sum\limits_{k=0}^{\eta-1} \left[\begin{array}{c} \left(\begin{array}{c} s_R[n+k]w_R[n+\eta+k] \\ +s_I[n+k]w_I[n+\eta+k] \end{array} \right) \\ + \left(\begin{array}{c} s_R[n+k]w_R[n+k] \\ +s_I[n+k]w_I[n+k] \end{array} \right) \end{array} \right] + \frac{1}{\eta} \sum\limits_{k=0}^{\eta-1} \left\{ \begin{array}{c} w_R[n+k]w_R[n+\eta+k] \\ +w_I[n+k]w_I[n+\eta+k] \end{array} \right\} \right) \\ \times \left(P + \frac{\sqrt{P}}{\eta} \sum\limits_{k=0}^{2\eta-1} \left(\begin{array}{c} s_R[n+k]w_R[n+k] \\ +s_I[n+k]w_I[n+k] \end{array} \right) + \frac{1}{2\eta} \sum\limits_{k=0}^{2\eta-1} \left(w_R^2[n+k] + w_I^2[n+k] \right) \right) \end{array} \right\} \\
&= \begin{array}{c} P^2 + \frac{P}{\eta^2} \sum\limits_{k=0}^{\eta-1} \left\{ \begin{array}{c} s_R^2[n+k]E(w_R^2[n+k]) \\ +s_I^2[n+k]E(w_R^2[n+k]) \end{array} \right\} + \frac{P}{\eta^2} \sum\limits_{k=0}^{\eta-1} \left\{ \begin{array}{c} s_R^2[n+k]E(w_R^2[n+\eta+k]) \\ +s_I^2[n+k]E(w_R^2[n+\eta+k]) \end{array} \right\} \\ + \frac{P}{2\eta} \sum\limits_{k=0}^{2\eta-1} \left\{ E(w_R^2[n+k]) + E(w_I^2[n+k]) \right\} \end{array} \\
&= P^2 + P\sigma^2/\eta + P\sigma^2
\end{aligned} \tag{3.21}$$

With the above analyses, we now calculate $E\left(r[n]\right)$ as

$$E\left(r[n]\right) = E\left(a_R[n]\right) - \rho E\left(b[n]\right) = (1-\rho)P - \rho\sigma^2 \tag{3.22}$$

Further, with (3.14), (3.19), and (3.21), we have

$$\begin{aligned} E\left(r^2[n]\right) &= E\left(a_R^2[n]\right) + \rho^2 E\left(b^2[n]\right) - 2\rho E\left(a_R[n]b[n]\right) \\ &= (1-\rho)^2 P^2 + \frac{(1-\rho)^2 + 2\eta\rho\left(\rho-1\right)}{\eta} P + \frac{1+\rho^2\left(2\eta+1\right)}{2\eta}\sigma^4 \end{aligned} \tag{3.23}$$

With (3.22), and (3.23), we have

$$Var(r[n]) = E\left(r^2[n]\right) - E^2\left(r[n]\right) = \frac{(1-\rho)^2}{\eta} P\sigma^2 + \frac{1+\rho^2}{2\eta}\sigma^4 \tag{3.24}$$

We now explain why $r[n]$ can be approximated as a Gaussian random variable. From (3.11) and (3.17), we see that both $b[n]$ and $a_R[n]$ are a sum of many noise terms. In a practical random-access system, the number of terms can be quite large (see the chapter below for numerical justifications). Hence, we can apply the Central Limit Theorem to approximate $b[n]$ and $a_R[n]$ as Gaussian variables [66]. Further considering that $r[n] = a_R[n] - \rho b[n]$, we approximate $r[n]$ as a Gaussian variable. Then with $E(r[n])$ and $Var(r[n])$ given in (3.22) and (3.24), we write the distribution of $r[n]$ as

$$r[n] \sim N\left((1-\rho)P - \rho\sigma^2,\ \frac{(1-\rho)^2}{\eta}P\sigma^2 + \frac{1+\rho^2}{2\eta}\sigma^4\right) \tag{3.25}$$

We can define things in terms of SNR by transforming $r[n]$ to $r[n]/\sigma^2$. After the transformation, we have

$$r[n] \sim N\left((1-\rho)\gamma - \rho\ ,\ \ \frac{(1-\rho)^2}{\eta}\gamma + \frac{1+\rho^2}{2\eta}\right) \tag{3.26}$$

where $\gamma = P/\sigma^2$ is the SNR of the studied antenna. In the rest of this chapter, unless stated otherwise, we mean the post-transformation $r[n]$ written as the function of γ when we mention $r[n]$.

Note that $r[n]$ in (3.26) represents the general setting. To analyze the missed-detection probability, we need to assume that there is a packet over the air (i.e., $\gamma \neq 0$). To analyze the false-alarm probability, on the other hand, we need to assume that there is no packet and there is only noise (i.e., $\gamma = 0$). For clear descriptions, let us distinguish $r[n]$ with "packet and noise input" and that with "noise input only" by subscript P and subscript N, respectively. Furthermore, we use $r[n]$ to represent general cases regardless of the input type. We have

$$r_P[n] \sim N\left((1-\rho)\gamma - \rho\ ,\ \frac{(1-\rho)^2}{\eta}\gamma + \frac{1+\rho^2}{2\eta}\right),\ \gamma \neq 0 \tag{3.27}$$

and

$$r_N[n] \sim N\left(-\rho\ ,\ \ \frac{1+\rho^2}{2\eta}\right) \tag{3.28}$$

To analyze false alarms and missed detections, we define z to be the normalized $r[n]$:

$$z = \frac{r[n] - E\left(r[n]\right)}{\sqrt{Var\left(r[n]\right)}}, \qquad z \sim N(0,1) \tag{3.29}$$

Note the normalization in (3.29) is the general case that works for both $r_P[n]$ and $r_N[n]$.

By the definition of missed detection (i.e., not claiming packet detection when there is a packet), we can write the missed-detection probability P_{MD}

as

$$
\begin{aligned}
P_{MD} &= \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{-\frac{E(r_P[n])}{\sqrt{Var(r_P[n])}}} e^{-\frac{z^2}{2}} dz = Q\left(\frac{E\left(r_P[n]\right)}{\sqrt{Var\left(r_P[n]\right)}}\right) \\
&= Q\left(\frac{\sqrt{\eta}\left((1-\rho)\gamma-\rho\right)}{\sqrt{(1-\rho)^2\gamma+(1+\rho^2)/2}}\right)
\end{aligned}
\tag{3.30}
$$

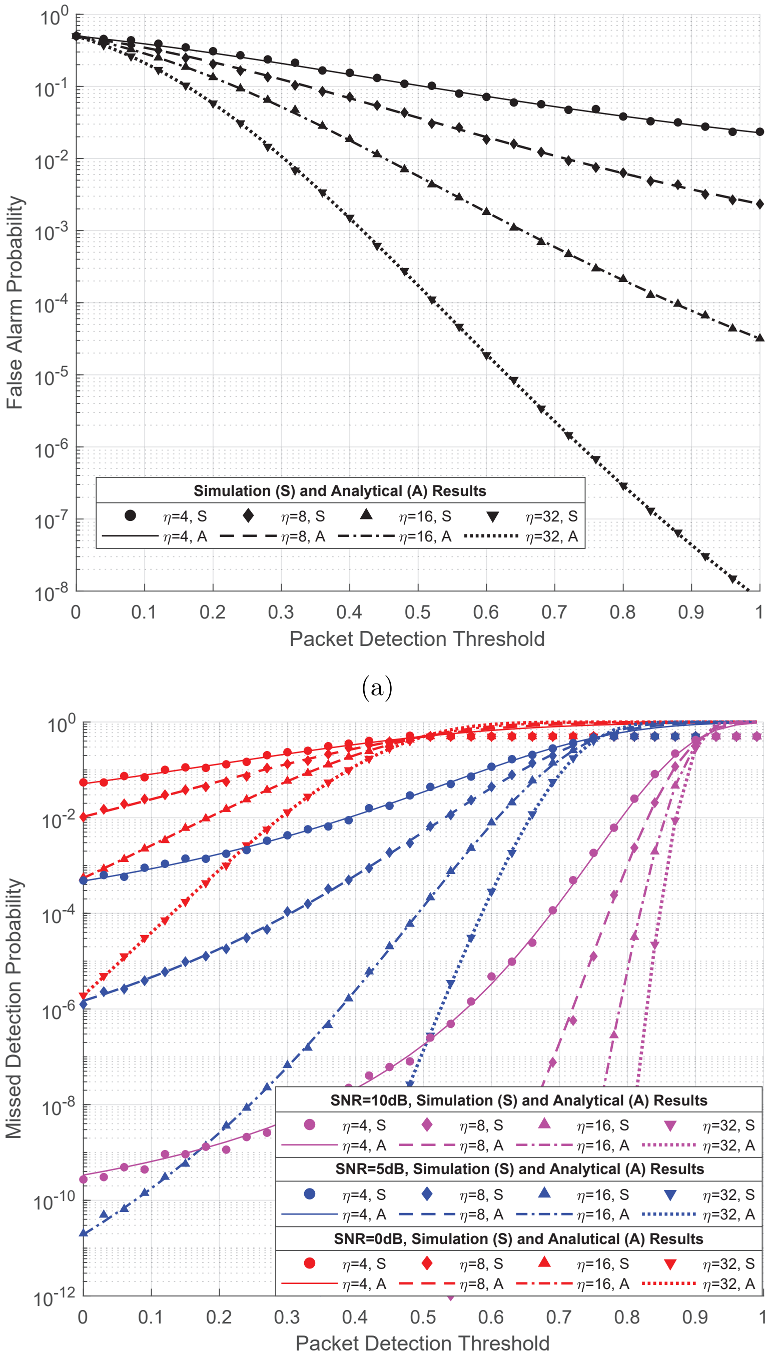
where $Q(.)$ is the well-known Q function [67].

By the definition of false alarm (i.e., claiming packet detection when there is no packet), we write the false-alarm probability P_{FA} as

$$
\begin{aligned}
P_{FA} &= \frac{1}{\sqrt{2\pi}} \int_{-\frac{E(r_N[n])}{\sqrt{Var(r_N[n])}}}^{\infty} e^{-\frac{x^2}{2}} dx = Q\left(-\frac{E\left(r_N[n]\right)}{\sqrt{Var\left(r_N[n]\right)}}\right) \\
&= Q\left(\sqrt{\frac{2\eta\rho^2}{1+\rho^2}}\right)
\end{aligned}
\tag{3.31}
$$

3.2.4 Validating Gaussian Assumption

In our analysis above, we make the Gaussian approximation on $r[n]$ to derive the closed-forms expressions of P_{MD} and P_{FA}. Here, we present simulation results that do not make the Gaussian approximation to validate the analytical results with the assumption. We can see from Fig. 3.4a and Fig. 3.4b that the simulation results annotated by markers well match the corresponding analytical curve when there are no less than 16 terms in the summation of $a_R[n]$ and $b[n]$, i.e., $\eta \geq 16$. In a practical random-access system, the length of an STS is typically no smaller than 16. For example, in IEEE 802.11, the total preamble length is 160 samples [68]. If we apply the two-STS setting as in this chapter, we have $\eta = 80$. Hence, we can confidently say that our Gaussian approximation on $r[n]$ is very precise on realistic systems, and the expressions of P_{MD} and P_{FA} we derived in (3.30) and (3.31) are trustworthy.



(a)

(b)

Figure 3.4: (a) Simulated and analytical results of false-alarm probability under various η and ρ settings. (b) Simulated and analytical results of missed-detection probability under various η, γ, and ρ settings.

3.3 Benchmarking RP-S&C With the Conventional S&C Algorithms

This chapter proposes a rigorous method to benchmark the performance of various packet detection schemes. We use this method to compare the performance of the conventional S&C algorithm with our RP-S&C algorithm.

Previous studies have evaluated packet detection schemes solely based on the missed-detection probability. For instance, [51] experimentally investigated several packet detection schemes for vehicular communication, but the authors simply concluded the superiority of one scheme over others based on the number of missed detected packets only, assuming the same threshold for all the schemes under consideration. However, the consideration of false-alarm probability exposes a fundamental flaw of this approach. In essence, a packet detection scheme can trade off between the probabilities of false alarm and missed detection by adjusting its threshold. Lowering the detection threshold reduces the probability of missed detection while simultaneously increasing the probability of false alarm. Thus, simply comparing missed detections without considering false alarms, or comparing false alarms without considering missed detections, is not reasonable.

A simple approach that simultaneously addresses both false alarms and missed detections is to construct a testing dataset and treat the packet detection process as a binary classification problem, where the goal is to determine whether a packet is present over the air. However, we emphasize that this approach heavily depends on the proportions of the presence and absence of packets in the testing dataset. For instance, if the dataset is collected in an idle environment with minimal traffic – i.e., where no CSMA packets are present for most of the time – then most of the data in the dataset correspond to the scenario where packets are absent. If scheme A is a lot more aggressive than scheme B in declaring the presence of a packet, then scheme A may not appear to be as good as scheme B in the classification problem. On the other hand, in the scenario where the traffic is heavy, most of the data correspond to the scenario where there is a packet, and scheme A may appear to be better than scheme B in the classification problem.

Pareto-superiority is a stricter form of comparison between schemes that are independent of the dataset. In particular, if we can show that scheme A is Pareto-superior to scheme B, then regardless of the the traffic situation, scheme A will have a better performing operating point than scheme B (i.e., that scheme A is better than scheme B is dataset independent). Appendix I expresses this notion in a mathematically rigorous manner.

We now elaborate the Pareto comparison. Suppose that we have two packet detection schemes, A and B. In general, we can adjust ρ_A and ρ_B to obtain the tradeoff curves for the operating points $\left(P_{FA}^{A}(\rho_A), P_{MD}^{A}(\rho_A)\right)$ and $\left(P_{FA}^{B}(\rho_B), P_{MD}^{B}(\rho_B)\right)$, respectively. To illustrate our point, in Fig. 3.5, we plot an example of $\left(P_{FA}^{A}(\rho_A), P_{MD}^{A}(\rho_A)\right)$ curve and an example of $\left(P_{FA}^{B}(\rho_B), P_{MD}^{B}(\rho_B)\right)$ curve for two fictitious schemes A and B. We now explain how we benchmark schemes A and B.

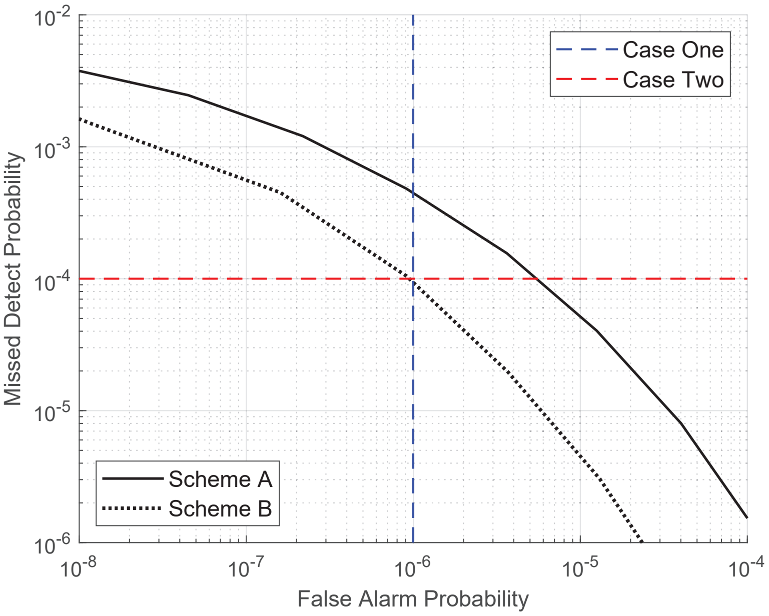


Figure 3.5: Two example MD-FA curves for illustrating our "Pareto comparison" benchmarking method.

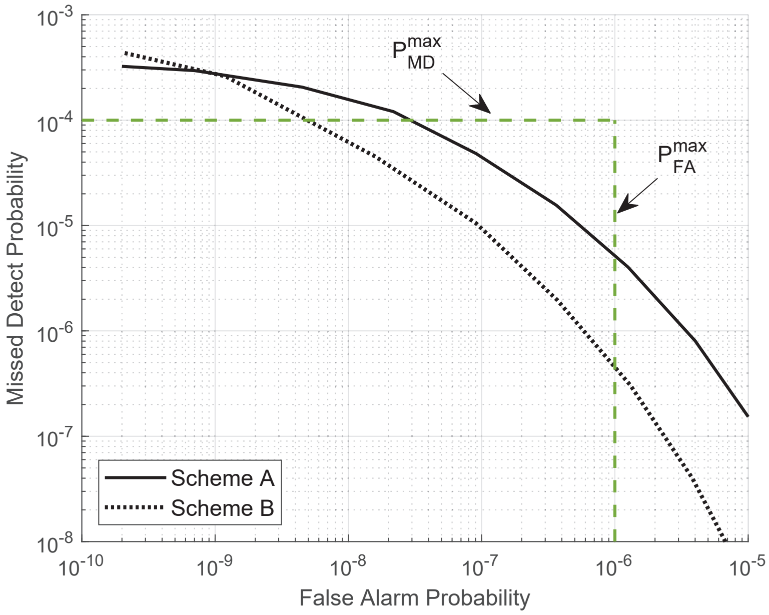
We can find the thresholds for schemes A and B, ρ_A and ρ_B, such that their false-alarm probabilities are equal. For example, in Case one of Fig. 3.5, we fix $P_{FA}^{A}(\rho_A) = P_{FA}^{B}(\rho_B) = 10^{-6}$. Note that ρ_A and ρ_B are not necessarily equal for the same false-alarm probability in both schemes. For Case one, we have $P_{MD}^{B}(\rho_B) < P_{MD}^{A}(\rho_A)$, and thus we say scheme B is superior to scheme A for this particular operating point. Alternatively, for Case two in the figure, we fix $P_{MD}^{A}(\rho_A) = P_{MD}^{B}(\rho_B) = 10^{-4}$ and observe that $P_{FA}^{B}(\rho_B) < P_{FA}^{A}(\rho_A)$, and thus again we say scheme B is superior to scheme A for this particular operating point. In general, in Fig. 3.5, scheme B is superior to scheme A in the Pareto-sense in that the overall P_{MD} versus P_{FA} curve (referred to as the MD-FA curve) of scheme B is lower than that of scheme A.

If the two curves crisscross each other, it is inconclusive as to which scheme is superior. However, as illustrated in Fig. 3.6a, if the curve of scheme B is consistently lower than that of scheme A within a specific region of interest (e.g., false-alarm probability not exceeding 10^{-6} and missed-detection probability not exceeding 10^{-4}), we can conclude that scheme B outperforms scheme A in that particular region (although the two curves may still crisscross outside the region of interest). Conversely, if the two curves intersect within the region of interest (as illustrated in Fig. 3.6b), we consider scheme A and scheme B to be comparable in that region, resulting in a “draw” in terms of benchmarking the two schemes.

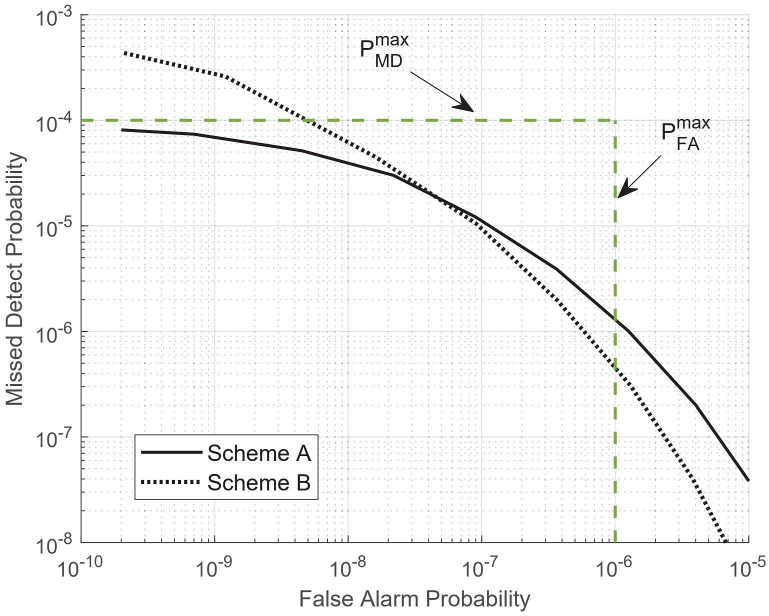
Given the above context, we now examine the performance of RP-S&C and conventional S&C. We assume a 0.2ppm/2ppm/5ppm oscillator offset in accordance with the state-of-the-art/typical/worst CFO condition that one may encounter in modern communication hardware.[6] Fig. 3.7 shows that the MD-FA curves of RP-S&C consistently lie below those of conventional S&C in various practical SNR and CFO settings, validating our statement that taking the real part of $a[n]$ is advantageous for packet detections.

---

[6] We conducted real-world experiments to test the CFOs of several commercial WiFi devices and examined the CFOs of a well-known open-source wireless channel dataset [51]. Our experiments revealed that the oscillator offsets of the tested hardware and the evaluated dataset are limited to a maximum of 2ppm. Thus, we take 2ppm as the typical oscillator offset. Moreover, we reviewed state-of-the-art research efforts published in top venues [55, 57] and found that the oscillator offsets reported no larger than 0.2ppm. Hence, we consider 0.2ppm to be the state-of-the-art oscillator offset. We assume a 5ppm oscillator offset as the worst-case scenario.

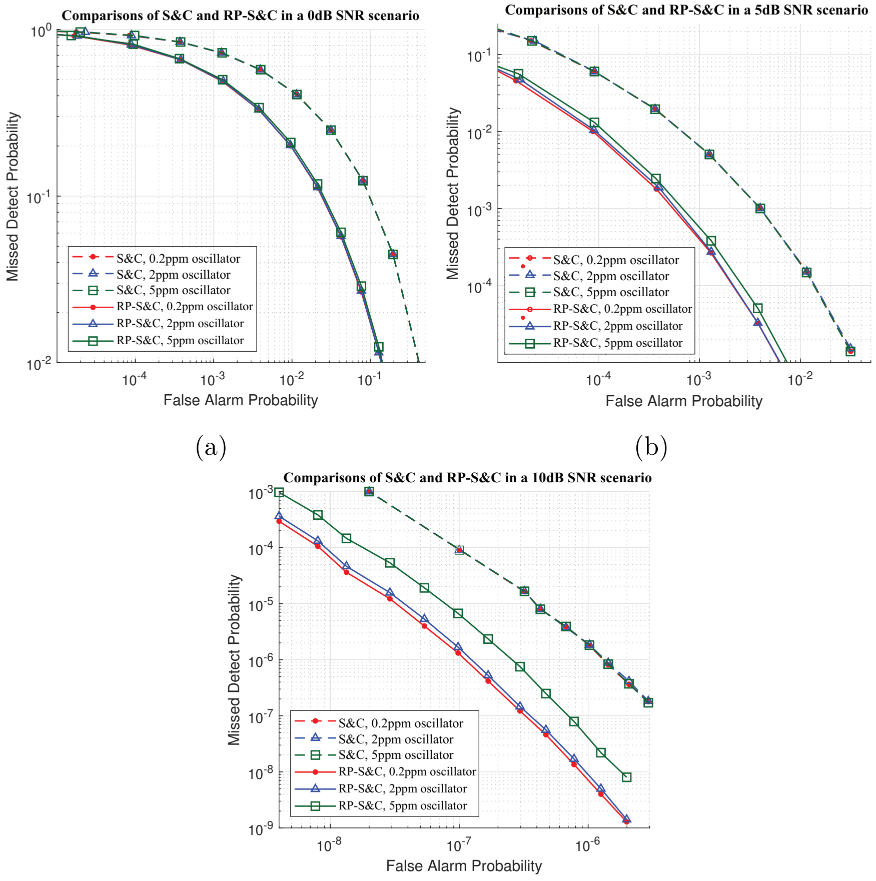


(a)



(b)

Figure 3.6: (a) An example to illustrate that one scheme is superior to the other within a specific region of interest. (b) An example to illustrate that one scheme is comparable with the other within a specific region of interest. The green dashed lines in the two figures demarcate the region of interest.



(a) (b)

(c)

Figure 3.7: RP-S&C versus conventional S&C under various CFO and SNR conditions.

3.4 RP-S&C for Multi-Antenna Packet Detection: Analysis and Optimization

3.4.1 Problem Formulation

Assume that there are N_R antennas in a receiver. Let us denote the $a_R[n]$, $b[n]$, and $r[n]$ of antenna j by $a_{R,j}[n]$, $b_j[n]$, and $r_j[n]$, respectively. We want to combine $r_j[n]$ with carefully chosen weights α_j so that the post-combined $r[n]$ yields good packet-detection performance in terms of false alarm or missed detection (or both). Let $r^M[n]$ represent the post-combined $r[n]$ in the multi-antenna case to distinguish it from $r[n]$ in the single-antenna case. The general expression of $r^M[n]$ is

$$r^M[n] = \sum_{j=1}^{N_R} \alpha_j r_j[n] = \sum_{j=1}^{N_R} \alpha_j \left(a_{R,j}[n] - \rho b_j[n]\right) \tag{3.32}$$

In the following, we investigate the optimal assignment for weight vector $\alpha = \{\alpha_1, ..., \alpha_{N_R}\}$. Recall from the discussion in Chapter 3.2.4 that both false alarm and missed detection are important aspects of a packet detection algorithm. For a given threshold ρ, the weight vector minimizing false-alarm probability is different from that minimizing missed-detection probability.

a. Minimizing False-Alarm Probability

Assume that there is no packet, as in (3.28), $r_{N,j}[n]$ can be approximated as a Gaussian random variable:

$$r_{N,j}[n] \sim N\left(-\rho\ ,\ \ \frac{1+\rho^2}{2\eta}\right) \tag{3.33}$$

Thus, we have

$$E\left(r_N^M[n]\right) = \sum_{j=1}^{N_R} \alpha_j E\left(r_{N,j}[n]\right) = -\rho \cdot \sum_{j=1}^{N_R} \alpha_j \tag{3.34}$$

and

$$Var\left(r_N^M[n]\right) = \sum_{j=1}^{N_R} \alpha_j^2 \cdot Var\left(r_{N,j}[n]\right) = \frac{1+\rho^2}{2\eta} \sum_{j=1}^{N_R} \alpha_j^2 \tag{3.35}$$

As in (3.31), the false-alarm probability in the multi-antenna case is given by

$$P_{FA}^M = Q\left(\sqrt{\frac{2\eta}{1+\rho^2}} \cdot \rho \cdot \sum_{j=1}^{N_R} \alpha_j \Bigg/ \sqrt{\sum_{j=1}^{N_R} \alpha_j^2}\right) \tag{3.36}$$

We note from (3.36) that, for any weight vector α, the weight vector scaled by a constant $c > 0$ yields the same P_{FA}^{M}. We can impose a normalization condition $\sum_{j=1}^{N_R} \alpha_j = 1$ without changing the outcome. Hence, we can formulate the optimization problem as

$$\begin{aligned} &\max f(\alpha) = \left(\sum_{j=1}^{N_R} \alpha_j^2 \right)^{-1}, \\ &\text{subject to } \sum_{j=1}^{N_R} \alpha_j = 1, \text{ and } \alpha_j \geq 0 \ \forall j \in \{1, ..., N_R\} \end{aligned} \tag{3.37}$$

b. Minimizing Missed-Detection Probability

Assume that there is a packet, as in (3.27), $r_{P,j}[n]$ can be approximated as a Gaussian random variable:

$$r_{P,j}[n] \sim N\left((1-\rho)\gamma_j - \rho \ , \ \ \frac{(1-\rho)^2}{\eta}\gamma_j + \frac{1+\rho^2}{2\eta} \right) \tag{3.38}$$

Thus, we have

$$E\left(r_P^M[n]\right) = \sum_{j=1}^{N_R} \alpha_j E\left(r_{P,j}[n]\right) = \sum_{j=1}^{N_R} \alpha_j \left[(1-\rho)\gamma_j - \rho\right] \tag{3.39}$$

and

$$\begin{aligned} Var\left(r_P^M[n]\right) &= \sum_{j=1}^{N_R} \alpha_j^2 \cdot Var\left(r_{P,j}[n]\right) \\ &= \sum_{j=1}^{N_R} \alpha_j^2 \left[\frac{(1-\rho)^2}{\eta}\gamma_j + \frac{1+\rho^2}{2\eta} \right] \end{aligned} \tag{3.40}$$

As in (3.30), the missed-detection probability is given by

$$\begin{aligned} P_{MD}^{M} &= Q\left(\frac{E\left(r_P^M[n]\right)}{\sqrt{Var\left(r_P^M[n]\right)}} \right) \\ &= Q\left(\sqrt{\eta} \frac{\sum_{j=1}^{N_R} \alpha_j \left[(1-\rho)\gamma_j - \rho\right]}{\sqrt{\sum_{j=1}^{N_R} \alpha_j^2 \left[(1-\rho)^2\gamma_j + (1+\rho^2)/2\right]}} \right) \end{aligned} \tag{3.41}$$

With (3.41), we impose a practical constraint[7] of $\gamma_i > \rho/(1-\rho)$ for every antenna so that we have $(1-\rho)\gamma_i - \rho > 0$, $\forall i \in \{1, ...N_R\}$, which means the numerator term within the Q function is always positive. With that, we can formulate the optimization problem as

$$\max g(\alpha) = \frac{\left(\sum_{j=1}^{N_R} \alpha_j \left[(1-\rho)\gamma_j - \rho\right]\right)^2}{\sum_{j=1}^{N_R} \alpha_j^2 \left[(1-\rho)^2\gamma_j + (1+\rho^2)/2\right]} \tag{3.42}$$

$$\text{subject to } \sum_{j=1}^{N_R} \alpha_j = 1, \text{ and } \alpha_j \geq 0 \ \forall j \in \{1, ..., N_R\}$$

Note from (3.37) and (3.42) that both minimizing false alarm and minimizing missed detection are subjected to the constraint of $\sum_{j=1}^{N_R} \alpha_j = 1$ and $\alpha_j \geq 0 \ \forall j \in \{1, ..., N_R\}$. In the rest of this chapter, we call $\alpha = \{\alpha_1, ..., \alpha_{N_R}\}$ a feasible weight vector only if it satisfies the constraint.

3.4.2 Optimal Weights for False Alarm (WFA) and Missed Detection (WMD)

Proposition 1. For a receiver with N_R antennas, equal-weight assignment $\alpha_j = 1/N_R$ to $r_{N,j}[n]$ yields the minimum P_{FA}^M.

Proof.

$$\underset{s.t. \sum \alpha_j = 1, \alpha_j \geq 0}{\underset{\alpha}{\arg\max}} \ f(\alpha) = \underset{s.t. \sum \alpha_j = 1, \alpha_j \geq 0}{\underset{\alpha}{\arg\min}} \sum_{j=1}^{N_R} \alpha_j^2 \tag{3.43}$$

It is easy to see that the answer to (3.43) is found by setting $\alpha_j = 1/N_R$ for all j. ■

Finding the optimal weights for P_{MD}^M is more challenging. Let us look at the derivative of $g(\alpha)$ over α_j:

$$\frac{\partial g(\alpha)}{\partial \alpha_j} \triangleq \frac{n(\alpha_j)}{d(\alpha_j)} \tag{3.44}$$

---

[7] We will later show that a typical ρ in practical multi-antenna systems is no larger than 0.5, which means $\rho/(1-\rho)$ is no larger than 0dB. If an antenna i has SNR γ_i of 0 dB or lower, it will not contribute much to packet detection and packet decoding, and we might as well omit it in both considerations. In other words, in this analysis, we assume that antennas with SNR of less than $\rho/(1-\rho)$ would not be used for packet detection purposes.

where the denominator $d(\alpha_j)$ is always positive, i.e.,

$$d(\alpha_j) = \left(\sum_{m=1}^{N_R} \alpha_m^2 \left[(1-\rho)^2 \gamma_m + \frac{1+\rho^2}{2} \right] \right)^2 > 0 \tag{3.45}$$

and the numerator $n(\alpha_j)$ is analyzed in (3.46):

$$\begin{aligned}
& n(\alpha_j) \\
&= \begin{array}{c} 2 \left(\sum\limits_{m=1}^{N_R} \alpha_m^2 \left[(1-\rho)^2 \gamma_m + \frac{1+\rho^2}{2} \right] \right) \left(\sum\limits_{m=1}^{N_R} \alpha_m \left[(1-\rho)\gamma_m - \rho \right] \right) \left[(1-\rho)\gamma_j - \rho \right] \\ - \left(\sum\limits_{m=1}^{N_R} \alpha_m \left[(1-\rho)\gamma_m - \rho \right] \right)^2 \cdot 2\alpha_j \left[(1-\rho)^2 \gamma_j + \frac{1+\rho^2}{2} \right] \end{array} \\
&= \underbrace{2 \left(\sum_{m=1}^{N_R} \alpha_m \left[(1-\rho)\gamma_m - \rho \right] \right)}_{Term\ A} \underbrace{\left\{ \begin{array}{l} \left(\left[(1-\rho)\gamma_j - \rho \right] \cdot \sum\limits_{m=1}^{N_R} \alpha_m^2 \left[(1-\rho)^2 \gamma_m + \frac{1+\rho^2}{2} \right] \right) \\ - \left[(1-\rho)^2 \gamma_j + \frac{1+\rho^2}{2} \right] \cdot \left(\sum\limits_{m=1}^{N_R} \alpha_m \alpha_j \left[(1-\rho)\gamma_m - \rho \right] \right) \end{array} \right\}}_{Term\ B} \\
&= \underbrace{2 \left(\sum_{m=1}^{N_R} \alpha_m \left[(1-\rho)\gamma_m - \rho \right] \right)}_{Term\ A} \underbrace{\left\{ \sum_{m=1}^{N_R} \alpha_m \left[\begin{array}{l} (\alpha_m - \alpha_j) \left((1-\rho)^3 \gamma_m \gamma_j - \frac{(1+\rho^2)\rho}{2} \right) \\ +(\alpha_m \gamma_j - \alpha_j \gamma_m) \frac{(1+\rho^2)(1-\rho)}{2} + \\ (\alpha_j \gamma_j - \alpha_m \gamma_m) \rho (1-\rho)^2 \end{array} \right] \right\}}_{Term\ B} \\
&= \underbrace{2 \left(\sum_{m=1}^{N_R} \alpha_m \left[(1-\rho)\gamma_m - \rho \right] \right)}_{Term\ A} \underbrace{\left\{ \begin{array}{l} \left[(1-\rho)\gamma_j - \rho \right] \cdot \sum\limits_{m \neq j} \alpha_m^2 \left[(1-\rho)^2 \gamma_m + \frac{1+\rho^2}{2} \right] \\ -\alpha_j \left[(1-\rho)^2 \gamma_j + \frac{1+\rho^2}{2} \right] \cdot \sum\limits_{m \neq j} \alpha_m \left[(1-\rho)\gamma_m - \rho \right] \end{array} \right\}}_{Term\ B}
\end{aligned} \tag{3.46}$$

In (3.46), we write $n(\alpha_j)$ as the product of term A and term B. In term A, we impose the practical constraint described in footnote seven once again, i.e., $(1-\rho)\gamma_m - \rho > 0$, $\forall m \in \{1, ...N_R\}$. With $\sum_{j=1}^{N_R} \alpha_j = 1$, $\alpha_j \geq 0$ and $(1-\rho)\gamma_m - \rho > 0$, we know that term A is positive. Within term B, the terms in which $m = j$ in the two summations cancel out each other, so we can exclude them in the summations and obtain the final form of term B as in the last line.

We note that for a locally optimal solution, we require $\frac{\partial g(\alpha)}{\partial \alpha_j} = \frac{n(\alpha_j)}{d(\alpha_j)} = 0$ for all $j \in \{1, ..., N_R\}$. Thus, term B should be zero and the following equation should hold:

$$\begin{aligned}&[(1-\rho)\gamma_j-\rho]\cdot\sum_{m\neq j}\alpha_m^2\left[(1-\rho)^2\gamma_m+\frac{1+\rho^2}{2}\right]\\&=\alpha_j\left[(1-\rho)^2\gamma_j+\frac{1+\rho^2}{2}\right]\cdot\sum_{m\neq j}\alpha_m\left[(1-\rho)\gamma_m-\rho\right]\end{aligned}\tag{3.47}$$

From (3.47), we have

$$\alpha_j=\frac{\sum\limits_{m\neq j}\alpha_m^2\left[(1-\rho)^2\gamma_m+(1+\rho^2)/2\right][(1-\rho)\gamma_j-\rho]}{\sum\limits_{m\neq j}\alpha_m\left[(1-\rho)^2\gamma_j+(1+\rho^2)/2\right][(1-\rho)\gamma_m-\rho]},\ j\in\{1,...,N_R\}\tag{3.48}$$

To investigate (3.48), we start from the simple two-antenna case, i.e., we only have antenna 1 and antenna 2. We have

$$\alpha_1=\frac{\alpha_2^2[(1-\rho)^2\gamma_2+(1+\rho^2)/2][(1-\rho)\gamma_1-\rho]}{\alpha_2[(1-\rho)^2\gamma_1+(1+\rho^2)/2][(1-\rho)\gamma_2-\rho]}\tag{3.49}$$

$$\alpha_2=\frac{\alpha_1^2[(1-\rho)^2\gamma_1+(1+\rho^2)/2][(1-\rho)\gamma_2-\rho]}{\alpha_1[(1-\rho)^2\gamma_2+(1+\rho^2)/2][(1-\rho)\gamma_1-\rho]}\tag{3.50}$$

The above gives

$$\begin{aligned}\frac{\alpha_1}{\alpha_2}&=\frac{[(1-\rho)^2\gamma_2+(1+\rho^2)/2][(1-\rho)\gamma_1-\rho]}{[(1-\rho)^2\gamma_1+(1+\rho^2)/2][(1-\rho)\gamma_2-\rho]}\\&=\frac{[(1-\rho)\gamma_1-\rho]}{[(1-\rho)^2\gamma_1+(1+\rho^2)/2]}\Bigg/\frac{[(1-\rho)\gamma_2-\rho]}{[(1-\rho)^2\gamma_2+(1+\rho^2)/2]}\end{aligned}\tag{3.51}$$

As in the above discussion, we impose the constraint of $\sum_{j=1}^{N_R}\alpha_j=1$. Thus, a feasible locally optimal solution for the two-antenna case is given by

$$\alpha_j=c\frac{[(1-\rho)\gamma_j-\rho]}{[(1-\rho)^2\gamma_j+(1+\rho^2)/2]}\tag{3.52}$$

where

$$j=1,2,\ \ c=\left(\sum_{m=1}^{2}\frac{[(1-\rho)\gamma_m-\rho]}{[(1-\rho)^2\gamma_m+(1+\rho^2)/2]}\right)^{-1}\tag{3.53}$$

We now extend the analysis to cases with more than two antennas. We shall see that the solution form of (3.52) is retained for the general case.

With the general expression of α_j given in (3.48), we can verify that a feasible locally optimal solution for a N_R-antenna case is

$$\alpha_j = c\frac{(1-\rho)\gamma_j - \rho}{(1-\rho)^2\gamma_j + (1+\rho^2)/2} \tag{3.54}$$

where

$$j = 1, 2, \ldots N_R,\ c = \left(\sum_{m=1}^{N_R} \frac{[(1-\rho)\gamma_m - \rho]}{[(1-\rho)^2\gamma_m + (1+\rho^2)/2]}\right)^{-1} \tag{3.55}$$

In the rest of this chapter, we denote the weight vector calculated according to (3.54) by α^o. We now prove that α^o is the unique solution that yields the global maximum $g(\alpha)$.

Proposition 2. If a feasible weight vector α does not satisfy (3.54), i.e., $\alpha \neq \alpha^o$, then α is non-optimal. Thus, α^o in (3.54) is the unique optimal solution to minimizing P_{MD} as per (3.47).

Proof. We prove that there is another solution α' that yields $g(\alpha') > g(\alpha)$.

We first note that it is not possible that $\alpha_j < \alpha_j^o$ for all $j \in \{1, \ldots, N_R\}$ or $\alpha_j > \alpha_j^o$ for all $j \in \{1, \ldots, N_R\}$ because that would mean $\sum_{j=1}^{N_R} \alpha_j < 1$ or $\sum_{j=1}^{N_R} \alpha_j > 1$. Thus, given that $\alpha \neq \alpha^o$ and that α is feasible (i.e., $\sum_{j=1}^{N_R} \alpha_j = 1$), there must be at least one k such that $\alpha_k/\alpha_k^o > 1$ and at least one i such that $\alpha_i/\alpha_i^o < 1$. Let us refer to $\alpha_j \big/ \alpha_j^o$ as the weight ratio of index j. In general, there could be multiple weight ratios of different indexes that attain the maximum, and multiple weight ratios of different indexes that attain the minimum. Let the respective sets be

$$\mathbf{K} = \left\{k : \frac{\alpha_k}{\alpha_k^o} = \max_j \frac{\alpha_j}{\alpha_j^o}\right\} \text{ and } \mathbf{I} = \left\{i : \frac{\alpha_i}{\alpha_i^o} = \min_j \frac{\alpha_j}{\alpha_j^o}\right\} \tag{3.56}$$

Now, consider a $k \in \mathbf{K}$ and an $i \in \mathbf{I}$. We have that

$$\alpha_k > \frac{\alpha_j}{\alpha_j^o}\alpha_k^o,\ \forall j \notin \mathbf{K} \quad \text{and} \quad \alpha_i < \frac{\alpha_j}{\alpha_j^o}\alpha_i^o,\ \forall j \notin \mathbf{I} \tag{3.57}$$

With (3.57), we now look back to (3.46) and prove that $n(\alpha_k) < 0$. For detailed proof, please refer to (3.58) below, where we obtain the last equality by substituting α_k^o and α_m^o in accordance with (3.54) into the second line.

$$\begin{aligned}
&n(\alpha_k)\\
&= \underbrace{2\left(\sum_{m=1}^{N_R}\alpha_m\left[(1-\rho)\gamma_m-\rho\right]\right)}_{Term\ A}\underbrace{\left\{\begin{array}{l}\left[(1-\rho)\gamma_k-\rho\right]\cdot\sum\limits_{m\neq k}\alpha_m^2\left[(1-\rho)^2\gamma_m+\frac{1+\rho^2}{2}\right]\\ -\alpha_k\left[(1-\rho)^2\gamma_k+\frac{1+\rho^2}{2}\right]\cdot\sum\limits_{m\neq k}\alpha_m\left[(1-\rho)\gamma_m-\rho\right]\end{array}\right\}}_{Term\ B}\\
&< \underbrace{2\left(\sum_{m=1}^{N_R}\alpha_m\left[(1-\rho)\gamma_m-\rho\right]\right)}_{Term\ A}\underbrace{\left\{\begin{array}{l}\left[(1-\rho)\gamma_k-\rho\right]\cdot\sum\limits_{m\neq k}\alpha_m^2\left[(1-\rho)^2\gamma_m+\frac{1+\rho^2}{2}\right]\\ -\left[(1-\rho)^2\gamma_k+\frac{1+\rho^2}{2}\right]\cdot\sum\limits_{m\neq k}\alpha_m^2\frac{\alpha_k^o}{\alpha_m^o}\left[(1-\rho)\gamma_m-\rho\right]\end{array}\right\}}_{Term\ B}\\
&= 0
\end{aligned}\tag{3.58}$$

Similar to the proof in (3.58), we can also show that $n(\alpha_i) > 0$. Thus, for an infinitesimally small $\varepsilon > 0$, we have that

$$\frac{\partial g(\alpha)}{\partial\alpha_i}\varepsilon-\frac{\partial g(\alpha)}{\partial\alpha_k}\varepsilon=\varepsilon\left[\frac{n(\alpha_i)}{d(\alpha_i)}-\frac{n(\alpha_k)}{d(\alpha_k)}\right]>0\tag{3.59}$$

Given that $g(\alpha)$ is twice differentiable in α_k and α_i, we can construct a feasible solution α' such that $g(\alpha') > g(\alpha)$ as follows:

$$\begin{cases}\alpha'_k=\alpha_k-\varepsilon/|\mathbf{K}|, & \forall k\in\mathbf{K}\\ \alpha'_i=\alpha_i+\varepsilon/|\mathbf{I}|, & \forall i\in\mathbf{I}\\ \alpha'_j=\alpha_j, & \forall j\in\mathbf{J}\end{cases}\tag{3.60}$$

where $\mathbf{J}=\{1,\dots,N_R\}-\mathbf{I}-\mathbf{K}$ is the complement set of $\mathbf{I}\cup\mathbf{K}$; $|\mathbf{I}|$ and $|\mathbf{K}|$ denote the cardinality of $\mathbf{I}$ and $\mathbf{K}$, respectively. ■

Remark 3. *Although the above proof is complete by itself, a question that we might ask is how large can* ε *be (i.e., it does not have to be infinitesimally small). By similar reasoning as in the proof, we note that as we increase* ε*, we would still have* $\partial g(\alpha')/\partial\alpha'_k<0$ *and* $\partial g(\alpha')/\partial\alpha'_i>0$ *provided that* $\alpha_k{}'>\left(\alpha_j\big/\alpha_j^o\right)\alpha_k^o$ *for all* $j\notin\mathbf{K}$ *and* $\alpha_i{}'<\left(\alpha_j\big/\alpha_j^o\right)\alpha_i^o$ *for all* $j\notin\mathbf{I}$.

Thus, we can increase ε *until either* $\alpha_k{}'=\left(\alpha_j\big/\alpha_j^o\right)\alpha_k^o$ *for some* $j\notin\mathbf{K}$ *(i.e.,* $\alpha_j\big/\alpha_j^o$ *is the second largest weight ratio here), or* $\alpha_i{}'=\left(\alpha_j\big/\alpha_j^o\right)\alpha_i^o$ *for some* $j\notin\mathbf{I}$ *(i.e.,* $\alpha_j\big/\alpha_j^o$ *is the second smallest weight ratio here), whichever equality is fulfilled first. In particular, we can set*

$$\varepsilon=\begin{cases}\min\left(\begin{array}{c}|\mathbf{K}|\left(\alpha_k-\max\limits_{j\notin\mathbf{K}}\left(\frac{\alpha_j}{\alpha_j^o}\alpha_k^o\right)\right)\\ ,|\mathbf{I}|\left(\min\limits_{j\notin\mathbf{I}}\left(\frac{\alpha_j}{\alpha_j^o}\alpha_i^o\right)-\alpha_i\right)\end{array}\right) & \text{if}\mathbf{J}\neq\varnothing\\ |\mathbf{K}|\left(\alpha_k-\alpha_k^o\right)=|\mathbf{I}|\left(\alpha_i^o-\alpha_i\right) & \text{if}\mathbf{J}=\varnothing\end{cases}\tag{3.61}$$

In fact, the above suggests an algorithmic way to march toward α^o *from an arbitrary feasible* α*. We perform (3.60) in accordance with (3.61). Then, with the new* α'*, the cardinality of the new* $\mathbf{I}$ *or the new* $\mathbf{K}$ *is enlarged. We repeat the procedure until we get an even better solution* α''*, The procedure is repeated until we reach* α^o*.*

3.5 RP-S&C for Multi-Antenna Packet Detection: Experiments and Discussions

Chapter 3.4.2 puts forth two weight-assignment solutions for the combination of $r[n]$: 1) WFA and 2) WMD. In a random-access network with co-located antennas, WFA and WMD have similar performance because co-located antennas have nearly the same SNR, resulting in similar weights for both WFA and WMD.[8] That is, WMD also results in roughly equal-weight assignments.

Packet detection in advanced wireless communication systems with distributed antennas, also known as distributed antenna systems (DAS), introduces different scenarios when comparing WFA and WMD because the SNRs at non-co-located antennas may vary widely. DAS offers two distinct advantages over conventional co-located antenna systems. First, co-located antennas suffer from a weakness in that the signal blockage between the transmitter antenna and the co-located receiver antennas results in no signal reception. In contrast, DAS allows for potential signal reception even if one receiver antenna is blocked, thanks to clear paths of other non-blocked antennas. Second, in DAS, the proximity between the transmitter and the nearest receiver antenna tends to be smaller than the distance between the transmitter and co-located receiver antennas, resulting in improved communication quality between the transmitter and the receiver. However, the distributed nature of DAS poses a new challenge in weight assignment. Antennas in DAS can be separated by tens to hundreds of wavelengths. This

---

[8] Signal at different co-located antennas may differ in phase, but it does not affect the calculation of $a_R[n]$ and $b[n]$. There is little SNR difference between co-located antennas. Hence, the weight assigned by WMD should be very close to that of WFA.

discrepancy in propagation length among transmit-receive antenna pairs leads to varying SNRs across the antennas. Given the varying SNRs, the benchmarking of WFA and WMD becomes an issue.

3.5.1 Implementation Issues

Implementation-wise, WFA is a simple and practical scheme, as it requires no additional system information except N_R, the number of antennas. As a result, WFA has lower implementation complexity, requires fewer computational resources, and consumes less signal processing time. In particular, WFA does not need knowledge of the antenna SNRs of the antennas since the weights do not depend on the SNRs.

For the false-alarm probability of WFA, we substitute $\alpha_j = 1/N_R$ into (3.34) and (3.35) and obtain

$$E\left(r_N^{WFA}[n]\right) = -\rho \quad \text{and} \quad Var\left(r_N^{WFA}[n]\right) = \frac{1+\rho^2}{2\eta N_R} \tag{3.62}$$

where the subscript N denotes pure noise input. We know from (3.62) that the false-alarm probability of WFA can be written as

$$P_{FA}^{WFA} = Q\left(-\frac{E\left(r_N^{WFA}[n]\right)}{\sqrt{Var\left(r_N^{WFA}[n]\right)}}\right) = Q\left(\sqrt{\frac{2\eta N_R \rho^2}{1+\rho^2}}\right) \tag{3.63}$$

For the missed-detection probability, we substitute $\alpha_j = 1/N_R$ into (3.39) and (3.40) and obtain

$$\begin{cases} E\left(r_P^{WFA}[n]\right) = \dfrac{1-\rho}{N_R}\displaystyle\sum_{j=1}^{N_R}\gamma_j - \rho \\ Var\left(r_P^{WFA}[n]\right) = \dfrac{(1-\rho)^2}{\eta N_R}\displaystyle\sum_{j=1}^{N_R}\gamma_j + \dfrac{1+\rho^2}{2\eta N_R} \end{cases} \tag{3.64}$$

where the subscript P denotes packet-plus-noise input. From (3.64), we have

$$\begin{aligned} P_{MD}^{WFA} &= Q\left(\frac{E\left(r_P^{WFA}[n]\right)}{\sqrt{Var\left(r_P^{WFA}[n]\right)}}\right) \\ &= Q\left(\sqrt{\eta}\cdot\left(\frac{1-\rho}{N_R}\sum_{j=1}^{N_R}\gamma_j - \rho\right)\Bigg/\sqrt{\frac{(1-\rho)^2}{N_R^2}\sum_{j=1}^{N_R}\gamma_j + \frac{1+\rho^2}{2N_R}}\right) \end{aligned} \tag{3.65}$$

Implementing WMD is more complex, as WMD requires knowledge of SNRs beforehand in order to calculate the weight of different antennas. However, obtaining the SNRs before packet detection is challenging, as accurate SNR estimation typically requires pilot-based signal processing, which is triggered by packet detection rather than preceding it. Thus, we have a “chicken-and-egg dilemma” where we need to know the precise SNRs before doing WMD, but typically SNR estimations happen after WMD.

A possible practical way to overcome the problem is to estimate the SNR using the preamble. Specifically, we can measure the power of the background noise when the receiver is idle. And then, as in (3.3), we calculate $b[n]$ for each antenna. We can coarsely estimate the SNR of an antenna with the noise power and the $b[n]$ of that antenna. However, this approach has limitations. First, the power of background noise may vary over time, but we only have the average noise power obtained during the idle period. Second, the weight assignment in this scheme is highly sensitive to interference. In shared spectrum environments, where most random-access systems are deployed, wireless interferences are common (say wireless packets from a Bluetooth device or a working microwave oven). These interferences may add to the preamble sequence and increase $b[n]$ (but such interference does not increase $a_R[n]$ or aid in the packet detection process), making the estimated SNR larger than its actual value. Consequently, this scheme may not be very reliable in practice.

We call the system with the coarse SNR estimation as practical WMD (P-WMD), and the hypothetical system with perfect *a priori* knowledge of SNRs as ideal WMD (I-WMD).

3.5.2 Benchmark WFA and WMD in two typical DAS scenarios

We now benchmark WFA and I-WMD/P-WMD with typical DAS scenarios (see Chapter 3.3 for the Pareto benchmarking method). In our benchmarking exercise, we set the maximum false alarm tolerance and missed detection tolerance at $P_{FA}^{\max} = 10^{-6}$ and $P_{MD}^{\max} = 10^{-4}$, respectively. That is, we are only interested in operating regions with false-alarm and missed-detection probabilities below these thresholds. We justify the tolerance settings in the following. In a practical DAS, it is reasonable to assume that a packet has no more than 1024 OFDM samples.[9] Suppose that we want the system to experience no more than one false alarm every 1000 packets on average. Then, the false-alarm probability should be no larger than $(1024 \times 1000)^{-1} \approx 10^{-6}$. As for the missed-detection probability, we take as reference the ultra-reliable low latency communication (URLLC) defined by 3rd generation partnership project (3GPP) that requires at least 99.99% successful packet decoding [70, 71], i.e., the transmission error tolerance is no larger than 0.01% or 10^{-4}. We assume a missed-detection tolerance commensurate with the reliability requirement of packet decoding, corresponding to a missing-detection probability of no more than 10^{-4}.

Furthermore, as in the previous analysis, we consider a two-STS preamble, with each STS having 16 samples (i.e., $\eta = 16$). We further assume that the timing offsets between different packets have been compensated prior to the application of WFA/I-WMD/P-WMD.[10]

We consider two typical DAS scenarios in the benchmark: the non-blocked scenario and the partially blocked scenario. In the non-blocked scenario, all transmit-receive pairs have clear propagation paths. In the partially blocked scenario, the propagation paths of some receive antennas are obstructed, leading to significantly lower SNRs than other antennas. We do not consider the case where all receiver antennas are blocked because it is unlikely to happen (since DAS is designed to avoid such situations). Furthermore, even if the rare occasion happens, improving packet-detection performance would be futile as the low SNRs in all antennas could prevent

---

[9] Recent WiFi standards, such as 802.11ax or future IEEE 802.11be, typically have a the packet length no larger than 1024 samples. Moreover, a recent technical trend is to achieve URLLC with short-packet communication (SPC), where a packet typically has no more than 50 bytes [69], making the packet length much shorter than 1024 samples.

[10] Sample misalignment can be an issue for DAS. Due to the path-length discrepancy between different transmit-receive antenna pairs, the samples collected at different antennas may not be aligned in time (i.e., different propagation latency at different antennas). The alignment of input samples is necessary before applying WFA/WMD in DAS. For preambles with two STSs, the $b[n]$ of each packet exhibits a sharp peak that indicates the clear starting position of the preamble (see Fig. 3.2). We utilize the peak for alignment.

successful packet decoding anyway.

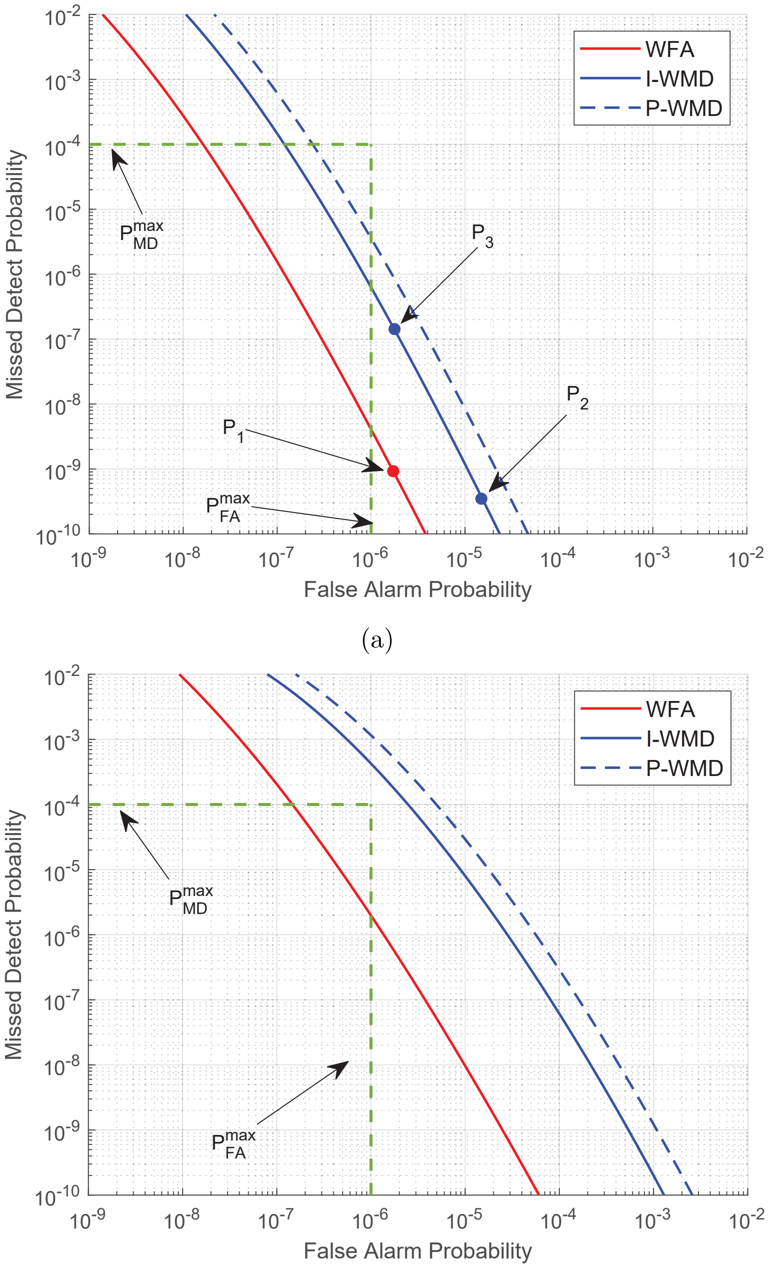
Table I gives two examples of the SNR conditions in the non-blocked scenario and the partially blocked scenario. Both examples consider four distributed antennas.

Table 3.1: SNR conditions of two typical DAS examples.

	Non-blocked DAS	Partially Blocked DAS
Ant.1	3.6118	0.2013
Ant.2	3.8903	3.5843
Ant.3	4.0338	3.3318
Ant.4	3.3649	4.2489

Fig. 3.8a presents the MD-FA curves of WFA, I-WMD, and P-WMD in the non-blocked scenario. It is clear that WFA outperforms the other two schemes in this example. The reader may wonder why the two WMD schemes turn out to have inferior missed-detection performance than WFA. The reason is simple: WMD is superior to WFA in terms of missed-detection performance only for a given fixed detection threshold ρ. However, WMD has a higher false-alarm probability for that fixed ρ. For example, in Fig. 3.8a, we fix $\rho = 0.45$ and highlight the corresponding (P_{FA}, P_{MD}) for WFA and I-WMD in points P_1 and P_2, respectively. As the figure shows, although P_2 has a lower missed-detection probability, its false-alarm probability is much higher than P_1. For the same false-alarm performance as WFA, I-WMD would have to raise its ρ, which in turn increases its missed-detection probability to the extent that it is worse than that of WFA (P_3 in Fig. 3.8a).

For the partially blocked scenario, as we can see from Table I, antenna one experiences a significantly low SNR due to blockage, while the other three antennas remain unaffected. Fig. 3.8b presents MD-FA curves of WFA, I-WMD, and P-WMD for this example. It is clear that WFA still outperforms the other two schemes in the partially blocked scenario.



(a)

(b)

Figure 3.8: (a) MD-FA curves of WFA/I-WMD/P-WMD in the non-blocked scenario. (b) MD-FA curves of WFA/I-WMD/P-WMD in the partially blocked scenario.

3.5.3 Benchmark Results in a Distributed Antenna Dataset

After examining the above two typical scenarios, we now proceed to a more general comparison between WFA and I-WMD/P-WMD. We use the same benchmark scheme as in the above discussion (including the same $P_{FA}^{\max}$ and $P_{MD}^{\max}$ settings) and conduct emulation experiments on DICHASUS [72], a massive open-source wireless channel dataset collected in industrial environments. To conserve space, we do not present the numerous MD-FA curves here.

We first give a general introduction of the dataset. The channel information in DICHASUS was measured using 32 software-defined radio (SDR) sensors and one transmitter that moves randomly in a factory. These 32 sensors were divided into four groups (Group A, B, C, and D), with each group comprising eight sensors located in one corner of the factory. Fig. 3.9 shows the layout of the factory and the locations of the antenna groups A, B, C, and D. The transmitter periodically transmits a reference packet that is known to every sensor. Upon receiving the reference packet, an SDR sensor compares it with the original version of the packet it knows *a priori* to obtain the precise channel information. The DICHASUS dataset encompasses in total 44,703 valid[11] DAS measurements collected in five different days, and each measurement has 32 pieces of channel information estimated by the 4x8 distributed sensors through the same reference packet at the same time.

We now give a look at the SNR information in DICHASUS. We denote an antenna by $Ant(i, j)$, where $i \in \{A, B, C, D\}$ and $j \in \{1, 2, 3, 4, 5, 6, 7, 8\}$. Each $Ant(i, j)$ has 44,703 SNR measurements, and we denote the kth SNR measurement by $\gamma_{i,j}(k)$. Unless stated otherwise, experiments and discussions below assume the original SNR values rather than the dB values.

With the above, we now study the SNR correlations of all 32 antennas

---

[11] In the data pre-processing stage, we discard a small portion of measurements that are obviously invalid (or even wrong). For example, SNRs of some antennas are unreadable (i.e., not a number, NaN) or smaller than 0dB. That may be caused by measurement (or data recording) errors during the data collection. Furthermore, studying cases with less than 0dB SNR is meaningless for our packet-detection research, as such cases will fail in packet decoding anyway.

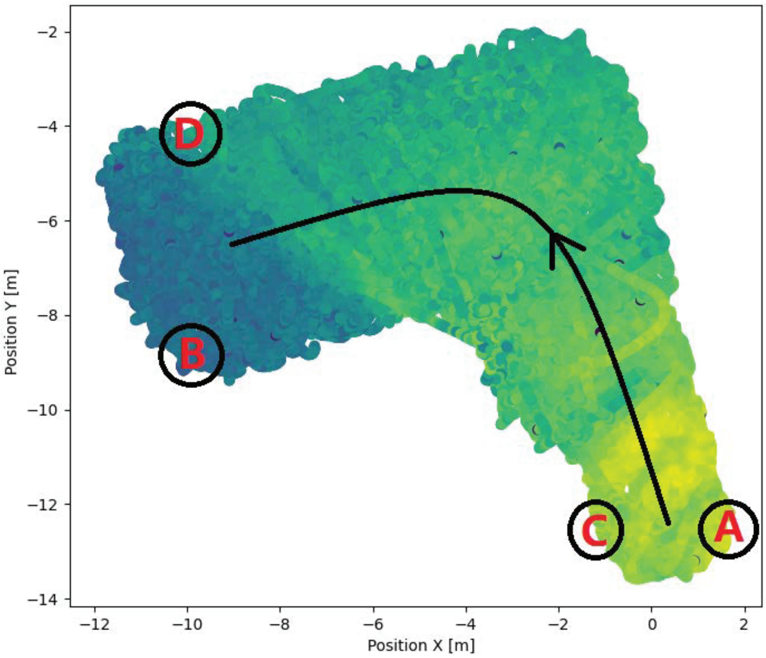


Figure 3.9: The location of Group A/B/C/D and an example of a moving transmitter.

and present the results in Fig. 3.10. Without loss of generality, we use $Ant(A,1)$ as a reference and calculate the correlation between $Ant(A,1)$ and $Ant(i,j)$ by

$$corr\left\{(A,1),(i,j)\right\} = \frac{\sum_k \left(\gamma_{A,1}(k) - \overline{\gamma_{A,1}}\right)\left(\gamma_{i,j}(k) - \overline{\gamma_{i,j}}\right)}{\sqrt{\sum_k \left(\gamma_{A,1}(k) - \overline{\gamma_{A,1}}\right)^2 \cdot \sum_k \left(\gamma_{i,j}(k) - \overline{\gamma_{i,j}}\right)^2}} \quad (3.66)$$

We have several observations from Fig. 3.10. First, SNRs of two co-located antennas are highly positively correlated. For example, Fig. 3.11a plots a 2-D scatter chart for $Ant(A,1)$ and $Ant(A,2)$, where a scatter point $(\gamma_{A,1}(k),\ \gamma_{A,2}(k))$ is plotted for each k. We see that the points roughly fall around the straight line $y = x$. We can obtain many similar 2-D scatter charts if we consider co-located antennas within the same group.

Second, if we look at $Ant(A,1)$ and antennas in Group D, we observe a weak and negative correlation between $Ant(A,1)$ and $Ant(D,j)$. Fig. 3.11b uses $Ant(D,6)$ as an example to illustrate the relationship. As we can see from the figure, the SNRs of $Ant(A,1)$ and $Ant(D,6)$ are generally

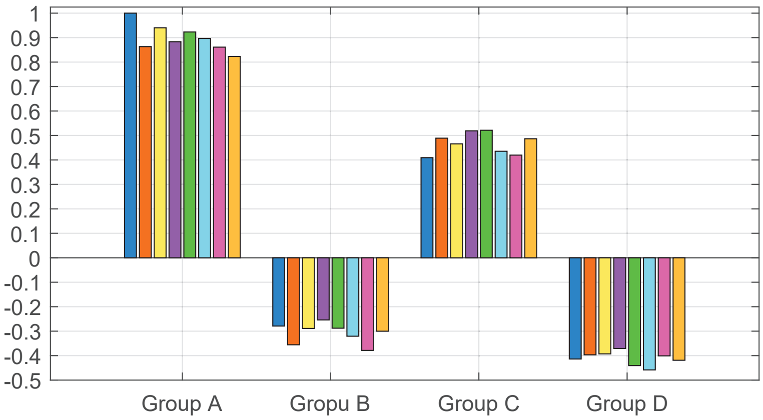
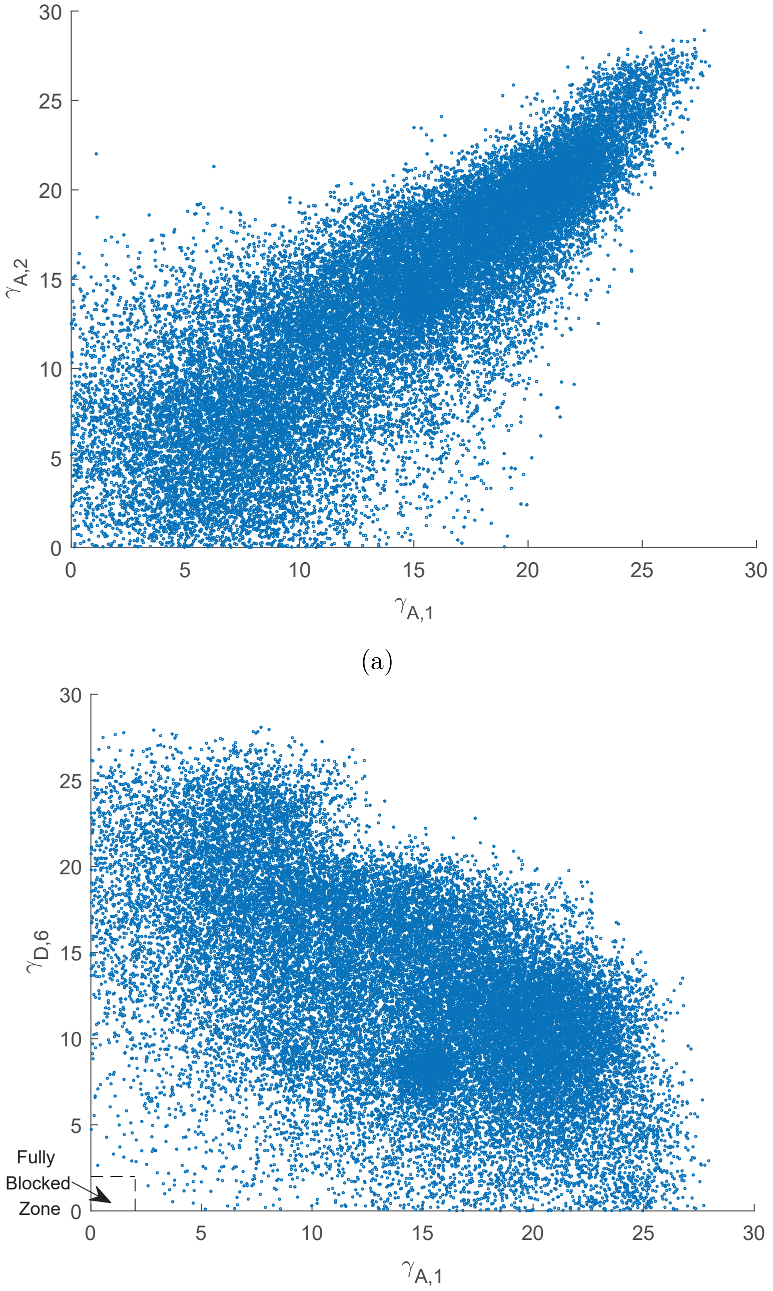


Figure 3.10: A visual illustration of SNR correlations between different antennas (using $Ant(A, 1)$ as a reference).

negatively correlated. This can be explained by the distributed nature of DAS: if one distributed antenna is weak, the other may still be strong. From Fig. 3.9, we see that Group A and Group D are located in two opposite corners of the factory. When the transmitter moves from one corner to the opposite corner (Fig. 3.9 also provides an example of the trace of a transmitter moving from Group A to Group D), we should observe a decrease in SNR for one and an increase in SNR for the other. Furthermore, thanks to this distributed nature of DAS, we see no fully blocked cases in Fig. 3.11b, i.e., at least one antenna has an SNR larger than 3dB.

We now elaborate on our experimental settings. In the first step, we construct a fixed two-STS preamble (with $\eta = 16$ in every STS). This preamble sequence is used consistently throughout the experiment. Since our objective is to benchmark WFA and I-WMD/P-WMD in general cases to see if one scheme consistently outperforms the other in practice, we emulate and test all channel measurements in DICHASUS. For the testing of each measurement, it is sufficient to represent a group of antennas with one or two elements, given that the SNRs of co-located antennas within the same group are highly positively correlated. Therefore, we randomly select two antennas from each group to simplify the emulation. After the random antenna selections, we emulate the corresponding 4x2 channels and transmit the fixed preamble sequence through these emulated channels. At the receiver side of



(a)

(b)

Figure 3.11: (a) The scatter plot of SNRs in $Ant(A,1)$ and $Ant(A,2)$. (b) The scatter plot of SNRs in $Ant(A,1)$ and $Ant(D,6)$. In (b), the “fully blocked zone” refers to the SNR region where both antennas have very low SNR. Both two figures use the original SNR values rather than the dB values.

the eight-antenna system, we apply WFA/I-WMD/P-WMD and record the benchmark result. After testing one measurement, we move on to the next measurement until we have completed testing all 44,703 measurements in the DICHASUS dataset.

During the benchmark process, we found that some antennas have significantly lower SNR than the other antennas when blockages occur. Let us make the following definition: if an antenna's SNR is lower than 3dB, we say the antenna is blocked and the measurement is a blocked case. Our analysis reveals that 14.47% of the emulated eight-antenna systems encounter varying degrees of blockage, while the remaining 85.53% of the emulated systems are non-blocked. Thanks to the reliability advantage of DAS, no fully blocked case (all antennas blocked) can be observed in the dataset.

Table 3.2: Benchmarking WFA and I-WMD/P-WMD with DICHASUS dataset.

	WFA v.s. I-WMD			WFA v.s. P-WMD		
	WFA win	Draw	I-WMD win	WFA win	Draw	P-WMD win
day 1	0.9218	0.0431	0.0351	0.9902	0.0048	0.0050
day 2	0.9002	0.0646	0.0352	0.9843	0.0079	0.0078
day 3	0.9104	0.0689	0.0207	0.9858	0.0101	0.0041
day 4	0.9169	0.0689	0.0142	0.9821	0.0107	0.0072
day 5	0.9251	0.0481	0.0268	0.9888	0.0058	0.0054
Avg.	0.9149	0.0587	0.0264	0.9862	0.0079	0.0059

Table II presents emulation results in detail, with 14.47% partially blocked tested cases and 85.53% non-blocked cases. In the table, each row corresponds to the benchmark conducted on a specific day. For each day, there are three pieces of data that record the percentage of instances where "WFA outperforms I-WMD/P-WMD in terms of packet detection", "Draw", and "I-WMD/P-WMD outperforms WFA" (please refer to Chapter 3.3 for the explanation of "Draw"). The last row of the table provides the average results across all 44,703 measurements.

Table II shows that WFA outperforms both P-WMD and I-WMD by a significant margin. On average, WFA surpasses its opponents in 91.49% of cases when competing against I-WMD and in 98.62% of cases when competing against P-WMD. Further, considering the draw cases, the percentage of WFA not losing is 97.36% when competing with I-WMD and 99.41% when competing with P-WMD.

Based on the above emulations, we recommend WFA as the desirable choice for a realistic DAS due to the following reasons:

1) (Superior packet-detection performance) WFA consistently outperforms both I-WMD and P-WMD in most cases, with an average percentage of 97.36% and 99.41%, respectively.

2) (Simplicity) WFA is much easier to implement, as it does not require SNR estimations or complex weight calculations. In contrast, I-WMD is not practical implementation-wise.

3) (Reliability) WFA is more reliable than P-WMD, as it is not sensitive to noise or interference. This is critical in practical applications where environmental factors may affect signal quality significantly.

3.6 Summary

This chapter provides a comprehensive treatment of packet detection for random access networks. The conventional S&C algorithm suffers from complex correlated noises in its packet-detection metric, making it difficult to analyze. To address this, we propose an analytical framework using compensated autocorrelation as the new metric for packet detection. Further experimental results demonstrate that taking the real part of the autocorrelation significantly enhances the performance of S&C.

By leveraging the analytical tractability of compensated autocorrelation, we obtain accurate closed-form expressions for false-alarm and missed-detection probabilities. These expressions provide a rigorous theoretical foundation for fair Pareto benchmarking of packet-detection schemes and extension of single-antenna packet detection schemes to multi-antenna packet detection schemes.

In particular, for multi-antenna detection, we can use the weighted sum of compensated autocorrelations at different antennas as the metric without sacrificing analytical rigor. This approach enables us to determine the best weights for minimizing the false-alarm probability (WFA) and the missed detection probability (WMD). Our investigation suggests that WFA is the preferred choice for practical application settings.

Overall, this chapter contributes to both the theory and practice of packet detection for random access networks. Our theoretical foundation provides insights on how to design packet detection schemes and how to compare and benchmark them in a rigorous manner in practical systems. Work conducted in this chapter has the potential to improve the performance of packet detection in random access networks and advance the field toward more efficient and reliable communication systems.

Chapter 4

LLM for FPGA-based SDR

4.1 Background

Wireless communication system prototyping on software-defined radio (SDR) enables real-world validation and is, therefore, a crucial aspect of wireless networking research [73]. Field-programmable gate arrays (FPGAs) have emerged as a popular SDR platform in recent years due to their significant reduction of processing latency compared to the conventional Universal Software Radio Peripheral (USRP) platform.

While FPGAs provide a versatile platform for system realization, their implementation poses a much steeper learning curve than conventional SDR platforms. Unlike traditional SDRs, for which high-level languages like C or Python are commonly used, FPGAs demand a more direct interaction with hardware through hardware description languages (HDLs) such as Verilog. The low-level programming, as well as the synthesis-friendly programming style required in HDL [22], necessitate a deep understanding of signal processing and internal hardware operations, prolonging the development cycle of SDR designs.

Recent advancements in large language models (LLMs) suggest a potential AI-assisted solution to accelerate HDL developments. Prior works have explored the use of LLMs for automated bug repairs [23], basic Verilog coding abilities [24], power-performance-area (PPA) optimization [25], and the realization of sequential logic circuits such as an 8-bit shift register [26].

These preliminary studies demonstrated LLMs' ability in generating simple HDL codes for small atomic computation tasks. But signal-processing algorithms in wireless SDRs are far more complex than these simple computational tasks demonstrated in prior works. Realizing wireless signal-processing algorithms requires highly parallel processing (for real-time operations) and involves many complex mathematical computations.

Whether LLMs can successfully implement intricate signal-processing algorithms remains an open question. This chapter investigates the implementation of the Fast Fourier Transform (FFT) algorithm as a case study to assess the ability of LLMs in realizing intricate signal-processing algorithms for wireless communication. We identify two primary challenges: 1) handling complex mathematical problems that require multi-step reasoning and 2) scheduling the execution of sub-modules within the complex hardware structure.

The first challenge arises when the target algorithm involves computation processes, for which numerical errors must be contained by proper scaling, such as in the implementation of the in FFT. This kind of task can be particularly difficult for LLMs due to their limited mathematical reasoning ability [74, 75]. To address this problem, we employ the chain-of-thought (CoT) prompting technique [76], which significantly enhances the mathematical performance of an LLM by showing it how a complex problem can be decomposed and solved in a step-by-step manner. This technique enables LLMs to break down the numerical calculation processes into smaller, more manageable sub-problems, making it easier for them to generate accurate solutions. Importantly, we employ a length-limit prompting strategy to reduce the error rate of the task.

The second challenge occurs when the HDL project contains numerous sub-modules to be executed with a complex timing sequence. For instance, in an FFT algorithm, butterfly computations within the same stage can be executed in parallel, while butterfly computations across different stages must be executed sequentially. Scheduling the execution of these computation tasks is challenging because it requires a deep understanding of the precedence relationships within the algorithm, which is beyond the capabilities of existing LLMs. To overcome this challenge, we leverage the iterative structure of the target FFT module and develop a novel prompting technique called iterative in-context learning (IICL). IICL automatically generates example question-answer (QA) pairs within the ICL framework in an iterative manner. With a series of QA pairs that progressively build upon each other, IICL enables the LLM to learn the complex timing sequences and precedence relationships required to schedule the execution of sub-modules effectively.

Through these technical advancements, we successfully generate the first LLM-written complex FPGA signal-processing module (specifically, a 64-point FFT). It is crucial to highlight that the CoT and IICL techniques are not limited to the implementation of the FFT module alone: they can be applied to a wide range of signal-processing modules. This versatility is due to the fact that complex mathematical problems and iterative structures are prevalent in wireless communication algorithms, making our techniques highly relevant and valuable in the field.

4.2 Related Works

Recent research has shown a growing interest in harnessing the capabilities of LLMs to assist researchers and engineers in hardware design. In [77], the authors employed GitHub Copilot to scrutinize the incidence rates of six types of Verilog bugs. Following this, [78] and [23] investigated the potential for automated bug repairs with LLMs' assistance. This trend is not limited to academia, as industry players such as RapidSilicon are promoting their upcoming LLM-assisted tool for hardware design, called RapidGPT [79].

Beyond assisting researchers and engineers, recent studies also show interest in replacing human HDL programmers with LLMs. Initial efforts in this direction were documented in [24], where a fine-tuned GPT-2 model was trained with synthetically generated Verilog snippets. However, the limited generalization ability to unfamiliar tasks was a notable shortcoming of [24]. Subsequent research [80] expanded on this concept by investigating various strategies for fine-tuning Verilog-writing models. More recently, two studies delved into LLMs' applications in chip design. The former, Chip-Chat [26], employed the latest LLM to design an 8-bit shift register chip; while the latter, ChipGPT [25], focused on the power-performance-area (PPA) optimization of an LLM-composed chip design.

In contrast to the aforementioned works, this thesis makes significant contributions from two key aspects.

In comparison to the first category of works [77, 78, 23, 79], the first contribution of this thesis involves using LLMs to assist FPGA development and offers a comprehensive study of the role of LLMs in facilitating the entire FPGA development process, beyond merely bug fixing. We investigate previously unexplored areas, including validation and maintenance issues, and our research includes the first LLM study that further refines a real-time communication system, OpenWiFi, that has already undergone rigorous validation and practical demonstrations previously.

In comparison to the second category of works [24, 80, 26, 25], our second contribution focuses on utilizing LLMs to write HDL code specifically for wireless communication hardware, involving significantly more complex signal-processing algorithms than those addressed in prior research. Earlier LLM-written HDL codes, as presented in [24] and [80], were confined to simple signal processing no harder than undergraduate-level assignments, as acknowledged by the authors in their subsequent work [26]. Although [26] and [25] advanced the complexity by tackling an 8-bit shift register and a multi-functional calculator, these tasks still fall short in complexity compared to communication algorithms such as FFT. As a result of tackling more demanding coding tasks, we encountered two challenges that had not been previously reported in prior works (see Chapter 4.3.2). Addressing these challenges necessitates the use of advanced methods, namely in-context learning (ICL) and chain-of-thought (CoT) prompting, which have never been considered in prior works [24, 80, 26, 25].

4.3 Existing Challenges

This chapter investigates the challenges LLMs may encounter when realizing complex signal-processing algorithms in wireless research, using the FFT algorithm as a case study. We consider the user requirement in Fig. 4.1 and present a typical Verilog realization that GPT-4 may generate in Fig. 4.2. Although the code looks structurally correct, it is not usable for two reasons: 1) it fails to generate precise expressions of the twiddle factors, and 2) it overlooks the scheduling of complex subtask executions.

This chapter analyzes the two problems in detail and tests the performance of different LLMs when facing these two problems to validate the commonality of the studied issue.

Initial prompt for generating the FFT module

Help me write an eight-point FFT module for my FPGA system in Verilog language. Here are details of my specifications: <***Function Descriptions***>

I also provide you with the following template for your reference: ***<Required Instantiation Template>***

Figure 4.1: A failed prompt for generating the FFT module.

4.3.1 Challenge One: Multi-step Reasoning

A twiddle factor W_N^k in FFT has a real/imaginary part no larger than one. For hardware processing, we need transform the real/imaginary of W_N^k to the corresponding binary for the digital circuitry to handle. Let us first take a look at how a human engineer might transform W_N^k into a 32-bit binary sequence (with a 16-bit imaginary part and a 16-bit real part) in a step-by-step manner, using W_8^1 as an example.

1. *Step 1*: express $W_8^1 = e^{-j(\pi/4)} = 0.7071 - 0.7071i$.

2. *Step 2*: multiply the real/imaginary part of W_8^1 by a scaling factor, constrained by the maximum value that can be represented by the number of bits allocated (in this case, 16 bits). We amplify $\operatorname{Re}\left(W_8^1\right)$ and $\operatorname{Im}\left(W_8^1\right)$ by $2^{15}-1$ so that $\operatorname{Re}\left(W_8^1\right) = 23169.5457$ and $\operatorname{Im}\left(W_8^1\right) = -23169.5457$.[1]

3. *Step 3*: round $\operatorname{Re}\left(W_8^1\right)$ and $\operatorname{Im}\left(W_8^1\right)$, so that $\operatorname{Re}\left(W_8^1\right) \approx 23170$ and $\operatorname{Im}\left(W_8^1\right) \approx -23170$.

4. *Step 4*: convert $\operatorname{Re}\left(W_8^1\right)$ and $\operatorname{Im}\left(W_8^1\right)$ to hexadecimal representations, i.e., “0x5A82” and “0xA57E”, respectively.

---

[1] At the butterfly computation’s output, we must shrink the multiplication output by the same scaling factor to maintain accuracy.

5. *Step 5*: concatenate the binary representations, using the higher/lower 16 bit as the imaginary/real part, respectively.

To quantitatively evaluate the difficulty of the above multi-step task for different LLMs, we tested the performance of the latest LLMs (including GPT-4, Claude-3, and Llama-2-70B) in the twiddle factor generation task. Fig. 4.3 presents the error rates of these models when generating twiddle factors for a 4/8/16-point FFT. The prompt we used in the experiment directly provides the tested LLM with information about the size of the desired FFT module and the length of the real/imaginary part.

According to our observation, typical errors made by these tested LLMs include 1) failure to transform W_N^k from the the exponential form to the ”real plus imaginary” form, 2) inability to correctly multiply W_N^k in Step 2[2], and 3) inability to correctly convert $Re\left(W_N^k\right)$ and $Im\left(W_N^k\right)$ to the corresponding hexadecimal representations. The experimental result shows that the precise generation of twiddle factors is a challenge even for these state-of-the-art models.

In Chapter 4.4, we show how we solve the multi-step reasoning problem and significantly enhance GPT-4’s performance in the twiddle factor generation task via CoT prompting. It is important to note that the CoT scheme can also be applied to other FPGA-based SDR development tasks that require complex multi-step reasoning, such as the parameter design for an IIR filter.

4.3.2 Challenge Two: Subtask Execution Scheduling

Many computational tasks can be divided into multiple simpler subtasks with parallel and precedence relationships. Fig. 4.4 presents the flow graph of an eight-point FFT to elucidate such relationships. The whole FFT computation can be decomposed into multiple butterfly computations. Butterfly

---

[2] We do not treat the twiddle factor generation as a failure if the scaling process in Step 2 is correct but non-optimal. The optimal scaling factor in Step 2 should be $2^{n-1}-1$ for an n-bit binary representation. A tested LLM may choose smaller scaling factors such as $2^{n-2}-1$ or $2^{n-3}-1$. We treat this as acceptable because the twiddle factor can still produce the correct result, although it is suboptimal in terms of accurate.

```
module fft_GPT_generate_8_point (
  input wire clk,
  input wire resetn,
  input wire enable,
  input wire [31:0] FFI_input_1,......,FFI_input_8,
  input wire input_valid,
  output wire [31:0] FFT_output_1,......,FFT_output_8,
  output wire output_valid
);
// Please fill in the Twiddle Factor
  localparam Twiddle_Factor_8_0=0x7FFF0000
  localparam Twiddle_Factor_8_1=0x5A82799A
  localparam Twiddle_Factor_8_2=0x00007FFF
  localparam Twiddle_Factor_8_3=0xA57E7A65

// Realize a butterfly computation submodule
  module butterfly (input wire [31:0] X1, input wire [31:0] X2,
            output wire [31:0] Y1, output wire [31:0] Y2);
    ......//Function Description of the butterfly computation
  endmodule
// Realize a complex multiplication submodule
  module comp_multi (input wire [31:0] X1, input wire [31:0] X2,
                     output wire [31:0] Y);
    ......// Function Description of the complex multiplication
  endmodule
//Internal signals
  wire [31:0] o11,o12,o13,o14,o15,o16,o17,o18;
  wire [31:0] o15_shift,o16_shift,o17_shift,o18_shift;
  wire [31:0] o21,o22,o23,o24,o25,o26,o27,o28;
  wire [31:0] o23_shift,o24_shift,o27_shift,o28_shift;
// Connections of submodules
  // Stage 1: butterfly
  butterfly butterfly_1(.X1(FFI_input_1), .X2(FFI_input_5), .Y1(o11), .Y2(o15));
  butterfly butterfly_2(.X1(FFI_input_2), .X2(FFI_input_6), .Y1(o12), .Y2(o16));
  butterfly butterfly_3(.X1(FFI_input_3), .X2(FFI_input_7), .Y1(o13), .Y2(o17));
  butterfly butterfly_4(.X1(FFI_input_4), .X2(FFI_input_8), .Y1(o14), .Y2(o18));
  // Stage 1: complex multiplication
  comp_multi comp_multi_1(.X1(o15), .X2(Twiddle_Factor_8_0), .Y(o15_shift));
  comp_multi comp_multi_2(.X1(o16), .X2(Twiddle_Factor_8_1), .Y(o16_shift));
  comp_multi comp_multi_3(.X1(o17), .X2(Twiddle_Factor_8_2), .Y(o17_shift));
  comp_multi comp_multi_4(.X1(o18), .X2(Twiddle_Factor_8_3), .Y(o18_shift));
  // Stage 2: butterfly
  butterfly butterfly_5(.X1(o11), .X2(o13), .Y1(o21), .Y2(o23));
  butterfly butterfly_6(.X1(o12), .X2(o14), .Y1(o22), .Y2(o24));
  butterfly butterfly_7(.X1(o15_shift), .X2(o17_shift), .Y1(o25), .Y2(o27));
  butterfly butterfly_8(.X1(o16_shift), .X2(o18_shift), .Y1(o26), .Y2(o28));
  //Stage 2: complex multiplication
  comp_multi comp_multi_5(.X1(o23), .X2(Twiddle_Factor_8_0), .Y(o23_shift));
  comp_multi comp_multi_6(.X1(o24), .X2(Twiddle_Factor_8_2), .Y(o24_shift));
  comp_multi comp_multi_7(.X1(o27), .X2(Twiddle_Factor_8_0), .Y(o27_shift));
  comp_multi comp_multi_8(.X1(o28), .X2(Twiddle_Factor_8_2), .Y(o28_shift));
  // Stage 3: butterfly
  butterfly butterfly_9(.X1(o21), .X2(o22),
                        .Y1(FFT_output_1), .Y2(FFT_output_2));
  butterfly butterfly_10(.X1(o23_shift), .X2(o24_shift),
                         .Y1(FFT_output_3), .Y2(FFT_output_4));
  butterfly butterfly_11(.X1(o25), .X2(o26),
                         .Y1(FFT_output_5), .Y2(FFT_output_6));
  butterfly butterfly_12(.X1(o27_shift), .X2(o28_shift),
                         .Y1(FFT_output_7), .Y2(FFT_output_8));
endmodule
```

Figure 4.2: A typical GPT4-written Verilog implementation for eight-point FFT.

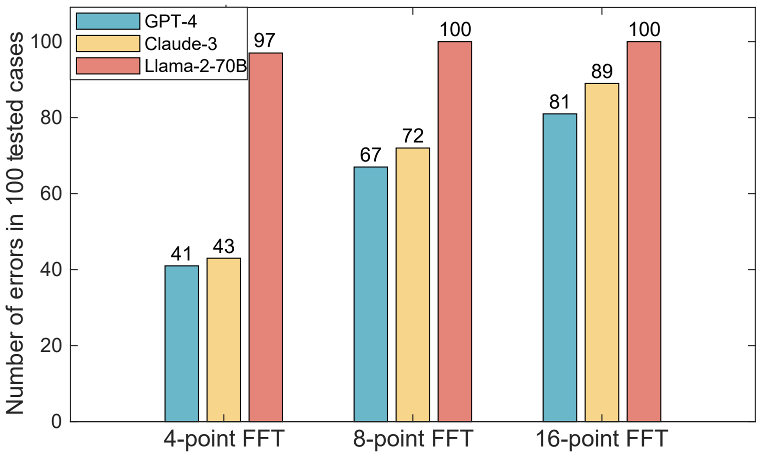


Figure 4.3: Error rates of different LLMs in twiddle factor generations.

computations from the same FFT stage can be executed concurrently to reduce latency, while those from successive stages can be parallelized only if the underlying precedence relationships are not violated.

The precedence relationships among butterfly computations necessitate careful subtask scheduling. The scheduling problem within FFT has been extensively studied in the past decades [81]. The most straightforward approach for precise subtask execution involves enabling signals and "output valid" signals (which become valid only when the associated subtask is completed and the output is meaningful). If the execution of subtask A depends on the completion of subtask B, we connect the "output valid" signal of B to the enable signal of A for execution management. Upon the completion of B, its output valid signal becomes positive, subsequently triggering the execution of A. The LLM-generated code in Fig. 4.2 shows no awareness of the need for task execution control: neither the basic method of using input enable/output valid signals nor more advanced methods like state machine-based control are presented. Our simulations on the GPT-written code in Fig. 4.2 found premature execution of subsequent subtasks before their predecessors, which causes erroneous outputs at the final stage.

To quantitatively evaluate the difficulty of subtask management for different LLMs, we tested the performance of three representative LLMs in scheduling the subtask execution within FFT modules of different sizes.

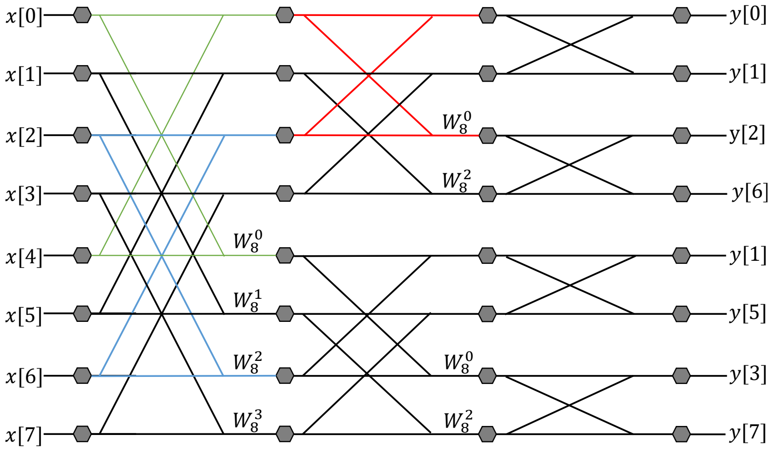


Figure 4.4: Precedence graphs of an eight-point FFT. The output of the green and the blue butterfly computations (in the first FFT stage) serve as the input for the red butterfly computation (in the second FFT stage). We cannot execute the red task until the blue and green tasks have been completed, indicating a precedence relationship between the red task and the blue/green task. We refer readers to Chapter 2.2.3 for detailed explanation of the parallel/precedence relationship.

Fig. 4.5 presents the error probabilities of scheduling butterfly computations within FFT modules of different sizes. The figure shows that the challenge of subtask scheduling commonly exists in these models.

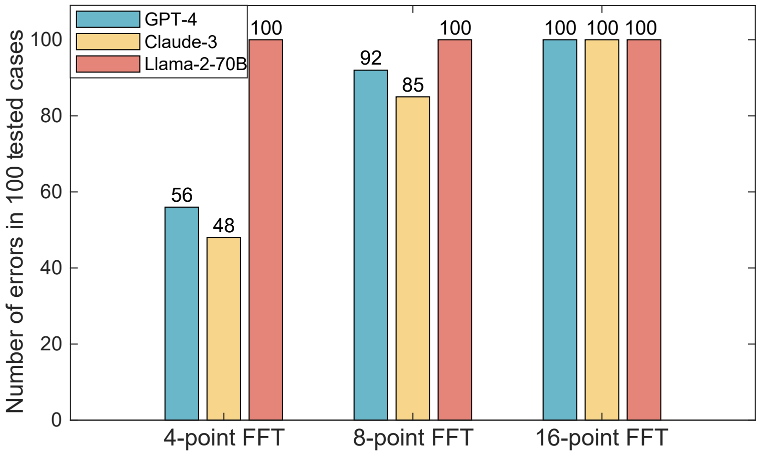


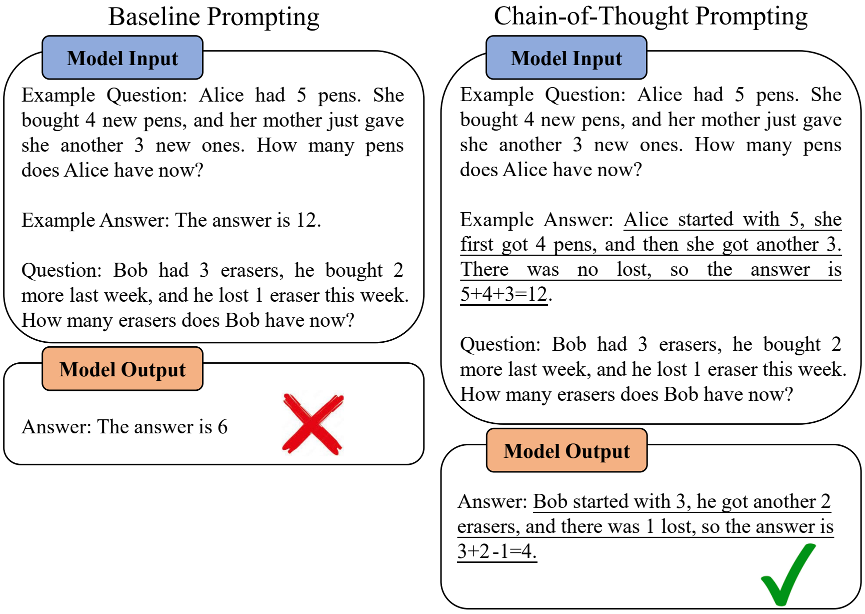
Figure 4.5: Error rates of different LLMs in scheduling butterfly computations within the FFT tasks.

Chapter 4.5 shows how we solve the scheduling problem and develop functionally correct FFT modules of different sizes with our novel IICL scheme. It is important to note that our scheme can also be applied to realize other iterative algorithms widely used in communication networks, such as the Gauss-Jordan Elimination [82] and the Newton method [83] commonly used in wireless signal processing.

4.4 Chain-of-Thought for Multi-Step Reasoning

Multi-step reasoning presents a significant challenge for language models. Recent studies have shown that guiding an LLM through a human's chain of thought by breaking down a complex problem into a series of intermediate reasoning steps can improve the model's performance in multi-step reasoning tasks [84]. This finding has inspired recent research interest in CoT prompting. Fig. 4.6 illustrates an example of the CoT scheme.

As described in Chapter 4.4, our twiddle factor generation task can be decomposed into five intermediate steps. Following the CoT framework, we



Example Question: Alice had 5 pens. She bought 4 new pens, and her mother just gave she another 3 new ones. How many pens does Alice have now?

Example Answer: The answer is 12.

Question: Bob had 3 erasers, he bought 2 more last week, and he lost 1 eraser this week. How many erasers does Bob have now?

Answer: The answer is 6

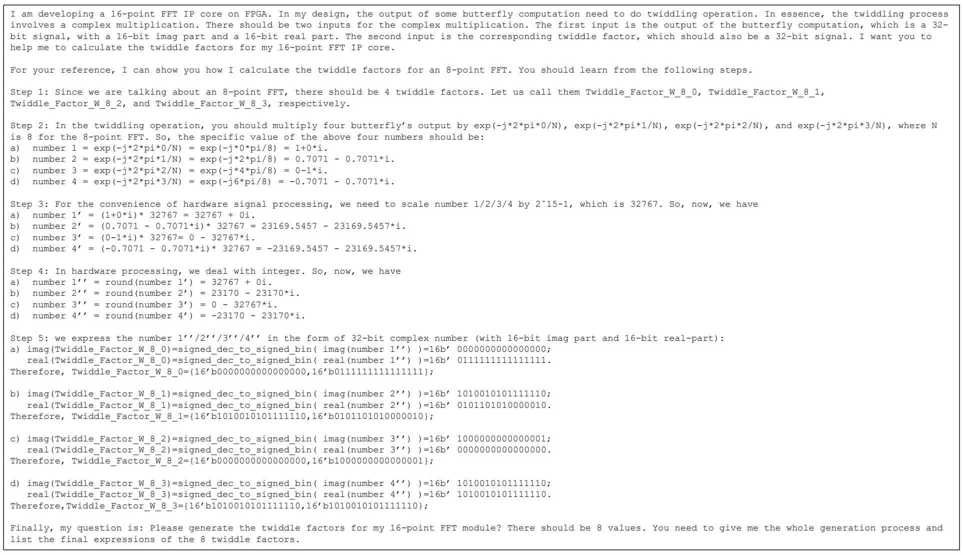
Example Question: Alice had 5 pens. She bought 4 new pens, and her mother just gave she another 3 new ones. How many pens does Alice have now?

Example Answer: Alice started with 5, she first got 4 pens, and then she got another 3. There was no lost, so the answer is 5+4+3=12.

Question: Bob had 3 erasers, he bought 2 more last week, and he lost 1 eraser this week. How many erasers does Bob have now?

Answer: Bob started with 3, he got another 2 erasers, and there was 1 lost, so the answer is 3+2-1=4.

Figure 4.6: A example to show how CoT enhance LLMs' performance in multi-step mathematical reasoning.



```
I am developing a 16-point FFT IP core on FPGA. In my design, the output of some butterfly computation need to do twiddling operation. In essence, the twiddling process
involves a complex multiplication. There should be two inputs for the complex multiplication. The first input is the output of the butterfly computation, which is a 32-
bit signal, with a 16-bit imag part and a 16-bit real part. The second input is the corresponding twiddle factor, which should also be a 32-bit signal. I want you to
help me to calculate the twiddle factors for my 16-point FFT IP core.

For your reference, I can show you how I calculate the twiddle factors for an 8-point FFT. You should learn from the following steps.

Step 1: Since we are talking about an 8-point FFT, there should be 4 twiddle factors. Let us call them Twiddle_Factor_W_8_0, Twiddle_Factor_W_8_1,
Twiddle_Factor_W_8_2, and Twiddle_Factor_W_8_3, respectively.

Step 2: In the twiddling operation, you should multiply four butterfly’s output by exp(-j*2*pi*0/N), exp(-j*2*pi*1/N), exp(-j*2*pi*2/N), and exp(-j*2*pi*3/N), where N
is 8 for the 8-point FFT. So, the specific value of the above four numbers should be:
a)  number 1 = exp(-j*2*pi*0/N) = exp(-j*0*pi/8) = 1+0*i.
b)  number 2 = exp(-j*2*pi*1/N) = exp(-j*2*pi/8) = 0.7071 - 0.7071*i.
c)  number 3 = exp(-j*2*pi*2/N) = exp(-j*4*pi/8) = 0-1*i.
d)  number 4 = exp(-j*2*pi*3/N) = exp(-j6*pi/8) = -0.7071 - 0.7071*i.

Step 3: For the convenience of hardware signal processing, we need to scale number 1/2/3/4 by 2^15-1, which is 32767. So, now, we have
a)  number 1’ = (1+0*i)* 32767 = 32767 + 0i.
b)  number 2’ = (0.7071 - 0.7071*i)* 32767 = 23169.5457 - 23169.5457*i.
c)  number 3’ = (0-1*i)* 32767= 0 - 32767*i.
d)  number 4’ = (-0.7071 - 0.7071*i)* 32767 = -23169.5457 - 23169.5457*i.

Step 4: In hardware processing, we deal with integer. So, now, we have
a)  number 1’’ = round(number 1’) = 32767 + 0i.
b)  number 2’’ = round(number 2’) = 23170 - 23170*i.
c)  number 3’’ = round(number 3’) = 0 - 32767*i.
d)  number 4’’ = round(number 4’) = -23170 - 23170*i.

Step 5: we express the number 1’’/2’’/3’’/4’’ in the form of 32-bit complex number (with 16-bit imag part and 16-bit real-part):
a) imag(Twiddle_Factor_W_8_0)=signed_dec_to_signed_bin( imag(number 1’’) )=16b’ 0000000000000000;
   real(Twiddle_Factor_W_8_0)=signed_dec_to_signed_bin( real(number 1’’) )=16b’ 0111111111111111.
Therefore, Twiddle_Factor_W_8_0={16’b0000000000000000,16’b0111111111111111};

b) imag(Twiddle_Factor_W_8_1)=signed_dec_to_signed_bin( imag(number 2’’) )=16b’ 1010010101111110;
   real(Twiddle_Factor_W_8_1)=signed_dec_to_signed_bin( real(number 2’’) )=16b’ 0101101010000010.
Therefore, Twiddle_Factor_W_8_1={16’b1010010101111110,16’b0101101010000010};

c) imag(Twiddle_Factor_W_8_2)=signed_dec_to_signed_bin( imag(number 3’’) )=16b’ 1000000000000001;
   real(Twiddle_Factor_W_8_2)=signed_dec_to_signed_bin( real(number 3’’) )=16b’ 0000000000000000.
Therefore, Twiddle_Factor_W_8_2={16’b0000000000000000,16’b1000000000000001};

d) imag(Twiddle_Factor_W_8_3)=signed_dec_to_signed_bin( imag(number 4’’) )=16b’ 1010010101111110;
   real(Twiddle_Factor_W_8_3)=signed_dec_to_signed_bin( real(number 4’’) )=16b’ 1010010101111110.
Therefore,Twiddle_Factor_W_8_3={16’b1010010101111110,16’b1010010101111110};

Finally, my question is: Please generate the twiddle factors for my 16-point FFT module? There should be 8 values. You need to give me the whole generation process and
list the final expressions of the 8 twiddle factors.
```

Figure 4.7: The CoT prompt we used to generate the 32-bit sequences for twiddle factors in a 16-point FFT.

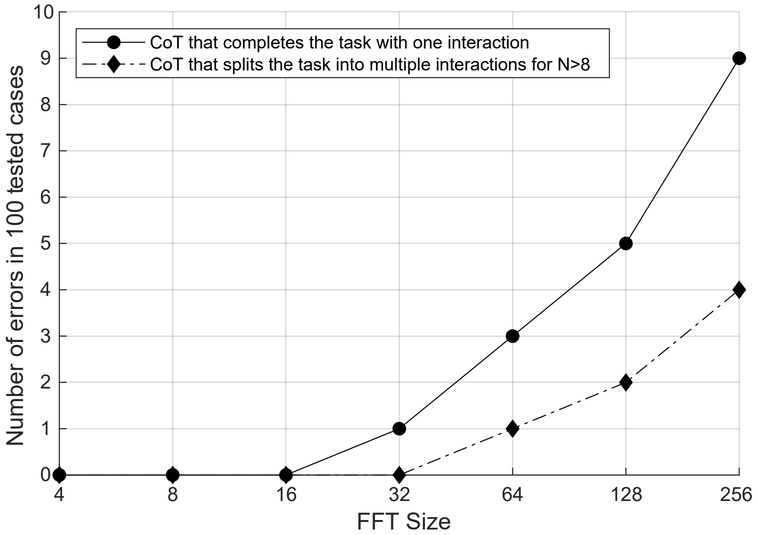


Figure 4.8: GPT-4 model's performance with the CoT prompting scheme.

developed a five-step task description for an eight-point FFT example. Using the chain of thought described in Fig. 4.7, we then instructed GPT-4 to generate twiddle factors for FFTs of various sizes (specifically, by changing the required FFT size in Fig. 6 to 4, 8, ..., 258). The solid line in Fig. 4.8 demonstrates that our CoT prompt significantly reduces GPT-4's error frequency in the twiddle factor generation process compared to its performance with the non-CoT prompt (represented by the blue bar in Fig. 4.5).

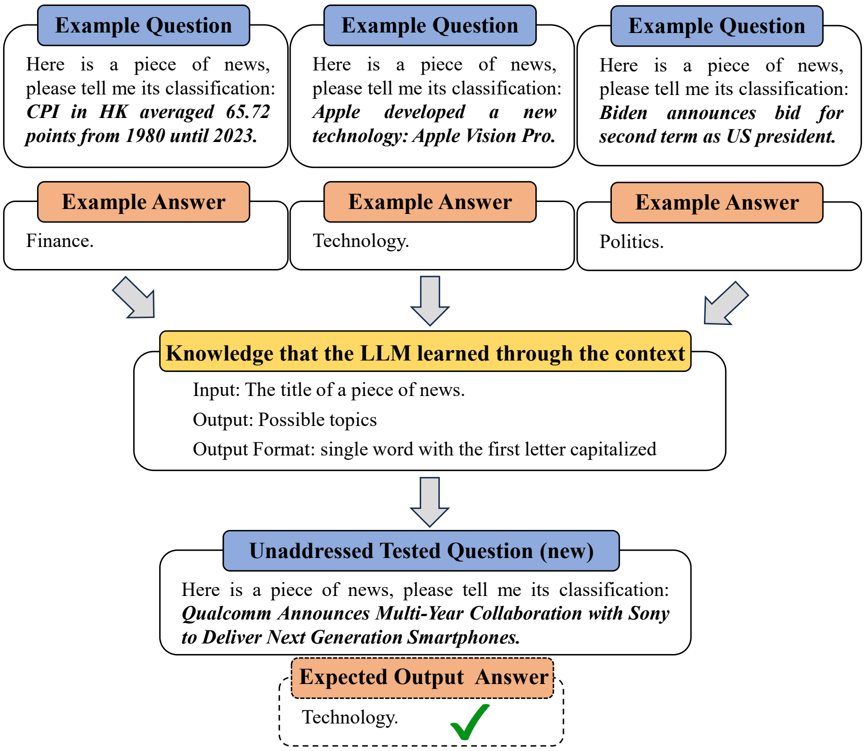
Moreover, we can observe from the curve that the task's error rate becomes higher with the increase of FFT size. This phenomenon can be explained with the attention-based mechanism of LLMs [85]. Specifically, when predicting the ith token of the answer, the transformer network takes all existing context, including the original prompt input and the $i-1$ token(s) already generated, as the model's input, which means the generated output also affects the model's attention weight [86, 87, 88]. Consequently, errors or inaccuracies in earlier processing can propagate and affect the correct chain of thought for subsequent tasks. As the FFT size increases, the LLM needs to generate more twiddle factors. If the earlier generation process introduces misleading interpretations of the chain of thought, even if the bias does not immediately cause erroneous twiddle factors, it increases the probability of making mistakes in later task executions.

We apply a length-limit strategy to adapt the original CoT prompt tuning scheme. Specifically, for an N-point FFT module, $N/2$ twiddle factors are required in the FFT computation. If $N/2$ is larger than a specific threshold, we divide the twiddle factor generation task into m groups, with each group containing $N/2m$ twiddle factors. Unlike a simplistic scheme that asks the LLM to generate all twiddle factors within one interaction, our length-limit CoT generates a limited number of twiddle factors at one time and re-informs the LLM of the five-step twiddle factor generation process at the beginning of each interaction. This scheme mitigates the long propagation of understanding bias given the shorter output texts and the repeated emphasis on the human's chain of thought. The dashed line in Fig. 4.8 illustrates the performance of the revised scheme (we set $N/2m = 8$ for $N/2 \geq 16$, i.e., we only consider eight twiddle factors for FFTs larger than 32 points), from which we observe that our scheme significantly reduces the error rate at the cost of more interactions.

4.5 Iterative In-context Learning for Task Schedule

Before presenting our IICL prompting scheme , let us first give a quick introduction to the conventional ICL method. The concept of ICL was popularized in [89], in which the authors introduced few-shot learning with the GPT-3 model using only a handful of examples. ICL differs from traditional machine learning scheme in that it does not require parameter optimization within the model [90]. In ICL, we give an LLM a prompt containing several question-answer (QA) pairs as examples to demonstrate how to complete a task. Following these pairs, a new question is appended to the prompt. The aim is for the LLM to analyze the previously given examples, extrapolate the underlying task, and provide an answer to this new question based on that learning context.

We illustrate ICL with the example in Fig. 4.9, in which a news classification task is described in the prompt. For the model's input, we provide several news titles and the corresponding topic classifications to create a series of QA pairs. We then present the LLM with a news title for which it must generate the topic classification. To correctly answer this question, the model must analyze the provided examples to understand the task. It learns the structure of the input (news titles), the range of possible outputs (possible news topics), the mapping from input to output (topic classification), and the formatting of the output (a single word with the first letter capitalized). With this understanding, the LLM generates the correct answer, i.e., “Technology”.



Here is a piece of news, please tell me its classification: ***CPI in HK averaged 65.72 points from 1980 until 2023.***

Here is a piece of news, please tell me its classification: ***Apple developed a new technology: Apple Vision Pro.***

Here is a piece of news, please tell me its classification: ***Biden announces bid for second term as US president.***

Finance.

Technology.

Politics.

Input: The title of a piece of news.
Output: Possible topics
Output Format: single word with the first letter capitalized

Here is a piece of news, please tell me its classification: ***Qualcomm Announces Multi-Year Collaboration with Sony to Deliver Next Generation Smartphones.***

Technology. ✓

Figure 4.9: Illustrations of conventional ICL prompting scheme.

We now return to the FFT flow graph in Fig. 4.4 to explicitly bring out the fundamentals of the iterative FFT algorithm. An N-point FFT has $log_2 N$ stages, where the $log_2 N - 1$ stages subsequent to stage 1 are essentially two parallel $N/2$-point FFTs. Therefore, as shown in Fig. 4.10, the flow graph can be simplified into two stages: Stage A and Stage B. The precedence/parallel relationships in the iterative flow graph are as follows: in stage A, the $N/2$ butterfly computations (including the complex multiplication of a twiddle factor for each butterfly computation) can be executed in parallel. The completion of Stage A then triggers the parallel execution of the two $N/2$-point FFTs in Stage B.

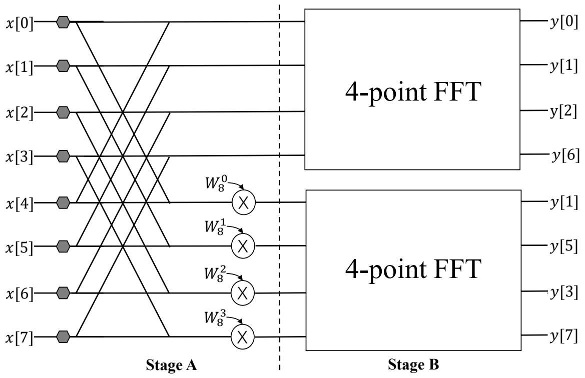


Figure 4.10: The iterative flow graph of an example eight-point FFT.

We now describe how we leveraged the iterative structure of FFT for IICL. With reference to step one of Fig. 4.12, we asked the LLM to prepare two simple yet frequently used IP cores for the following FFT design: butterfly computation and complex multiplication. We then induce the LLM to produce successively larger FFTs from previously generated smaller FFTs in an iterative manner. First, in the second step of Fig. 4.12, we showed GPT-4 a simple two-point FFT example with the corresponding implementation template, and then we asked the LLM to produce a four-point FFT IP core with a required template. To help the LLM better complete the task, we also gave it suggestions that highlight the precedence/parallel relationship delineated in Fig. 4.10 by demonstrating how to connect the “enable” and “output valid” signals of different subtasks (see Fig.4.11 for the suggestion we give to the LLM, which includes an example to illustrate the connection of “enable” and “output valid” signals). Thanks to the simplicity of the four-point FFT and our hints, GPT-4 succeeded in completing the task at the end of the second step. The resulting code, together with the prompt used in step two, was then used in the third step as a ICL example, which reinforced the LLM’s knowledge in FFT design by 1) outlining how to connect inputs and outputs of subtasks according to the iterative flow graph, and 2) showing how to control the execution of subtasks with “enable” and “output valid” signals.

```
I am writing a four-point DIF-FFT on FPGA. You can use the following IP cores
to build the target four-point FT IP core.

Here is the template of butterfly computation IP Core,
  butterfly_computation butterfly_inst(
    .clk(),
    .reset(),
    .enable(),
    .in1(), .in2(),
    .done(),
    .out1(), .out2()
  );

And here is the template of the two-point FFT IP Core,
  fft_2_point fft_2_point_inst(
    .clk(),
    .reset(),
    .enable(),
    .x1(), .x2(),
    .done(),
    .y1(), .y2()
  );
And here is the template of complex multiplication IP Core,
  complex_multiplication complex_multiplication_inst(
    .clk(),
    .reset(),
    .enable(),
    .in1(), .in2(),
    .done(),
    .out()
  );

I also give you the template of the target four-point FFT IP Core,
  fft_4_point fft_4_point_stage_2_1(
    .clk(),
    .reset(),
    .enable(),
    .x1(), .x2(), .x3(), .x4(),
    .y1(), .y2(), .y3(), .y4(),
    .done()
  );

Further, I also have some suggestions for you.

First, you don't need to calculate the expressions of twiddle factors.
I will fill in the value myself later.

Second, let me tell you the major spirit in developing the 4-point FFT.
In the first stage of FFT, you have two substages called Stage 1a and 1b.
In stage 1a, you use twobutterfly_computation modules; and then in stage 1b,
you use two twiddle models. In the second stage, you are facing a two
two-point FFT problem, so you just use two fft_2_point and solve the problem.
```

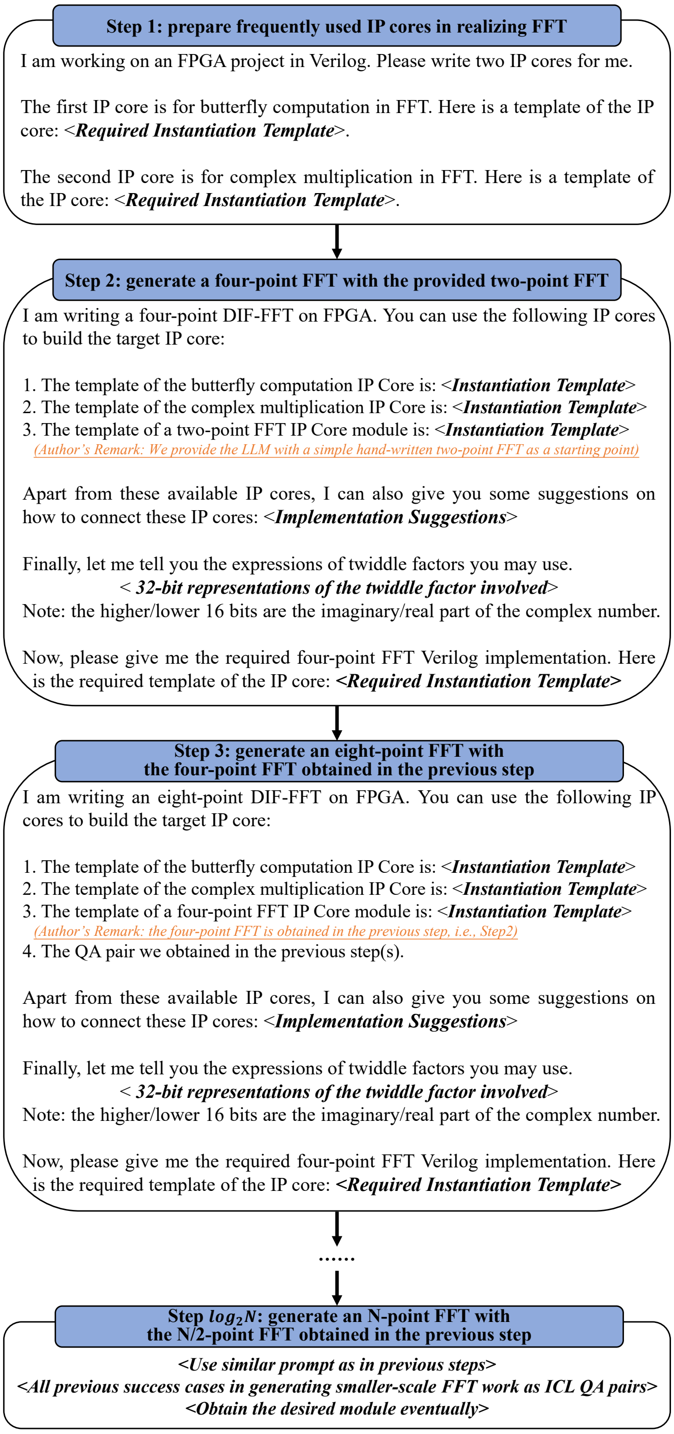
Figure 4.11: Implementation suggestion given to the LLM, which is related to the connection of “enable” and “output valid” signals. Here, we are telling the LLM how to build a four-point FFT by properly connecting the two two-point FFT modules provided.

The next few steps shown in Fig. 4.12 describe how to induce the LLM to extrapolate from the 4-point FFT to the 8-point FFT, and more generally, the iterative process to generate an N-point FFT. For step n $(3 \leq n \leq log_2 N)$, we provided the LLM with available IP cores (i.e., butterfly computation, complex multiplication, and the 2^{n-1}-point FFT IP core generated in Step $n-1$) and the implementation template of the desired 2^n-point FFT IP core, and we repeated the hints we previously used in the third step. As a result, GPT-4 could always generate the 2^n-point FFT desired in step n. The resulting 2^n-point IP core could further enhance the LLM's understanding of the task when it was used to form a new ICL example for the IICL process in step $n+1$. We followed the above operation iteratively until we obtained the desired N-point FFT design.

Following the above iterative ICL prompting scheme, we succeeded in generating a 64-point FFT design for concept proving. Compared to those non-IICL assisted attempts recorded in Chapter 4.3.2, the LLM-written Verilog code resulting from our IICL scheme demonstrates a significant improvement: from Fig. 4.5, we see that no FFT modules larger than eight points can be successfully generated even with those state-of-the-art LLMs. To our best knowledge, this is the first LLM-written complex signal processing model ever reported in wireless communication studies.

4.6 Summary

This chapter presents a pioneering exploration of utilizing LLMs to expedite the development of complex signal-processing algorithms for FPGA-empowered SDR systems, using the well-known FFT algorithm as an example. We tackled two primary challenges encountered in the implementation of FFT in the Verilog language: 1) complex mathematical calculations and 2) the efficient scheduling of sub-module executions. The CoT prompting technique we used for the first problem significantly enhanced the LLM's capacity for multi-step mathematical reasoning, enabling it to handle the intricate task of twiddle factor generation. Meanwhile, the IICL approach we developed for the second problem effectively taught the model to schedule the execution of sub-modules by conceptualizing the scheduling task itera-



Step 1: prepare frequently used IP cores in realizing FFT

I am working on an FPGA project in Verilog. Please write two IP cores for me.

The first IP core is for butterfly computation in FFT. Here is a template of the IP core: <***Required Instantiation Template***>.

The second IP core is for complex multiplication in FFT. Here is a template of the IP core: <***Required Instantiation Template***>.

Step 2: generate a four-point FFT with the provided two-point FFT

I am writing a four-point DIF-FFT on FPGA. You can use the following IP cores to build the target IP core:

1. The template of the butterfly computation IP Core is: <***Instantiation Template***>
2. The template of the complex multiplication IP Core is: <***Instantiation Template***>
3. The template of a two-point FFT IP Core module is: <***Instantiation Template***>

(Author's Remark: We provide the LLM with a simple hand-written two-point FFT as a starting point)

Apart from these available IP cores, I can also give you some suggestions on how to connect these IP cores: <***Implementation Suggestions***>

Finally, let me tell you the expressions of twiddle factors you may use.
< ***32-bit representations of the twiddle factor involved***>
Note: the higher/lower 16 bits are the imaginary/real part of the complex number.

Now, please give me the required four-point FFT Verilog implementation. Here is the required template of the IP core: ***<Required Instantiation Template>***

Step 3: generate an eight-point FFT with the four-point FFT obtained in the previous step

I am writing an eight-point DIF-FFT on FPGA. You can use the following IP cores to build the target IP core:

1. The template of the butterfly computation IP Core is: <***Instantiation Template***>
2. The template of the complex multiplication IP Core is: <***Instantiation Template***>
3. The template of a four-point FFT IP Core module is: <***Instantiation Template***>

(Author's Remark: the four-point FFT is obtained in the previous step, i.e., Step2)

4. The QA pair we obtained in the previous step(s).

Apart from these available IP cores, I can also give you some suggestions on how to connect these IP cores: <***Implementation Suggestions***>

Finally, let me tell you the expressions of twiddle factors you may use.
< ***32-bit representations of the twiddle factor involved***>
Note: the higher/lower 16 bits are the imaginary/real part of the complex number.

Now, please give me the required four-point FFT Verilog implementation. Here is the required template of the IP core: ***<Required Instantiation Template>***

……

Step log_2N: generate an N-point FFT with the N/2-point FFT obtained in the previous step

<Use similar prompt as in previous steps>
<All previous success cases in generating smaller-scale FFT work as ICL QA pairs>
<Obtain the desired module eventually>

Figure 4.12: An illustration of IICL prompting in our FFT generation task.

tively. The successful deployment of these techniques not only underscores the advanced capabilities of current LLMs in handling complex algorithmic generation tasks on SDR platforms, but also illustrates a substantial leap toward their practical applications in real-world engineering problems, particularly in the telecommunications domain. Future studies could explore the scalability of our prompting scheme to other complex systems and algorithms encountered in wireless communications.

Chapter 5

Conclusion

5.1 Conclusion Remarks

This thesis presents a comprehensive study on advanced multiple access techniques for Ultra-Reliable Low-Latency Communication (URLLC), addressing both theoretical challenges and practical implementation issues in next-generation wireless systems. The research is structured around three primary contributions.

First, the thesis reduces the computational latency of a novel Interleaved Frequency Division Multiple Access (IFDMA) transceiver design newly proposed, referred to as the compact IFDMA transceiver. With a novel FFT pruning methodology and a multi-priority scheduling (MPS) algorithm, the work significantly reduces the latency and resource consumption in hardware implementations. Extensive experiments demonstrate that the proposed MPS-FFT framework achieves near-optimal scheduling, leading to substantial improvements in both computation time and processor utilization compared to conventional solutions.

Second, the thesis addresses reliability issues in Carrier Sense Multiple Access (CSMA) based systems, particularly focusing on the largely-overlooked problem of packet detection. By introducing the compensated autocorrelation metric and establishing a rigorous analytical framework, the work enables closed-form analysis and fair benchmarking of different detection algorithms. The proposed real-part-only S&C (RP-S&C) algorithm is shown to offer a better packet-detection performance, especially under practical oscillator conditions. This work also extends the RP-S&C algorithm from the single-antenna scenario to the multi-antenna scenario.

Third, the thesis explores the application of large language models (LLMs) for accelerating the development of FPGA-based URLLC systems. Two innovative prompting strategies, i.e., Chain-of-Thought (CoT) reasoning and Iterative In-Context Learning (IICL), are proposed to enhance the ability of LLMs in generating high-quality Verilog code for complex signal processing modules. Case studies on FFT implementations validate the effectiveness and potential of AI-assisted hardware design.

Overall, the findings of this thesis contribute valuable insights and practical tools for the design of high-performance, reliable URLLC systems, laying a foundation for future research in both intelligent wireless networking and AI-driven hardware automation.

5.2 Future Works

First, for the novel multiple access mechanisms proposed in this work, future research can focus on deploying and validating these techniques in real industrial production environments. Compared to cellular networks, which often involve high deployment and operational costs, there is growing interest in exploring cost-effective platforms, such as WiFi-based systems, for achieving reliable and low-latency communication. These platforms leverage unlicensed spectrum, which not only reduces costs but also provides flexibility in deployment, making them particularly attractive for industrial applications. Instead of targeting the ultra-reliability and low-latency communication (URLLC) defined in high-cost 5G networks, future research can focus on fulfilling the latency and reliability requirement of a communication network with a relatively small cost, referred to as the very reliable and low-latency communication (VRLLC). VRLLC represents a more practical approach for systems built on low-cost platforms, where the stringent requirements of URLLC are adjusted to balance reliability, latency, and cost. Achieving VRLLC could lead to the development of scalable, efficient, and adaptive industrial production systems based on improved multiple access techniques.

Second, for the application of LLMs in assisting SDR design, future research could explore the development of next-generation intelligent agents capable of autonomously generating, optimizing, and adapting communication algorithms for various hardware platforms. Beyond SDR, LLMs could also be applied to other areas of wireless communication, such as adaptive protocol optimization, multimodal network management, and real-time decision-making for complex, decentralized wireless systems. These advancements would further demonstrate the transformative potential of AI-driven methodologies in accelerating the design and deployment of advanced communication systems.

Together, these two research directions represent a pathway toward bridging the gap between theoretical innovation and practical deployment, driving the evolution of intelligent, reliable, and low-latency systems for next-generation industrial and communication networks.

Appendix A

Isomorphism of FFT Precedence Graphs

In graph theory, isomorphism is an equivalence relation between graphs. The isomorphism of directed graphs $G_1\left(V_1, E_1\right)$ and $G_2\left(V_2, E_2\right)$ is defined as a bijection $h: V_1 \rightarrow V_2$ between the vertex sets V_1 and V_2 , and a bijection $g: E_1 \rightarrow E_2$ between the edge sets E_1 and E_2 such that there is a directed edge $e_1 \in E_1$ from $v_1 \in V_1$ to $v_2 \in V_1$ if and only if there is a directed edge $g(e_1) \in E_2$ from $h(v_1) \in V_2$ to $h(v_2) \in V_2$. As a particular class of directed graphs with precedence relationships, the FFT precedence graphs also have isomorphism relationships.

Recall from Chapter 2 that we denote a bin allocation by an ordered list $\{S_0, S_1, ...S_{R-1}\}$, where S_r is the set of bins allocated to stream $r (r = 0, 1, ..., R-1)$. We can further simplify the notation of bin allocation to $(s_0, s_1, ...s_{R-1})$, where s_r is the number of bins stream r has. For example, $S_0 = \{0, 1, 2, 3\}$ means that we allocate bin #0 to bin #3 to IFDMA stream 0, and $s_0 = 4$ indicates that we allocate the first four bins (which is also bin #0 to bin #3) to stream 0.

As discussed in Chapter 2, we assume that the bins are fully allocated, i.e., $\sum\limits_{r=0}^{R-1} s_r = 2^n$. We also assume that the number of bins for each IFDMA stream is power-of-two. Accordingly, a bin allocation can be represented by the leaf nodes of a power-of-two splitting tree (abbreviated as splitting tree henceforth). For example, Fig. A.1 presents the FFT

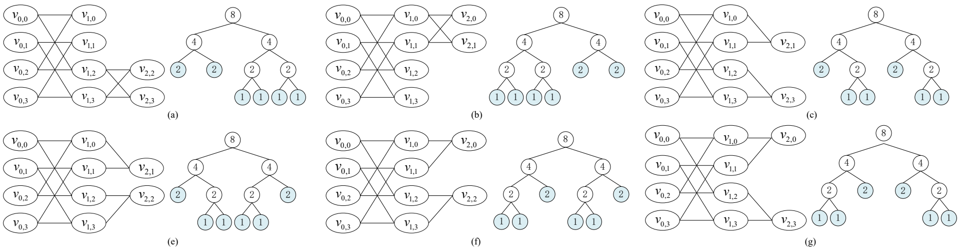


Figure A.1: FFT precedence graphs and corresponding splitting trees for bin allocation $(2,2,1,1,1,1)$, $(1,1,1,1,2,2)$, $(2,1,1,2,1,1)$, $(2,1,1,1,1,2)$, $(1,1,2,1,1,2)$, and $(1,1,2,2,1,1)$.

precedence graphs and their corresponding splitting trees for bin allocations $(2,2,1,1,1,1)$, $(1,1,1,1,2,2)$, $(2,1,1,2,1,1)$, $(2,1,1,1,1,2)$, $(1,1,2,1,1,2)$, and $(1,1,2,1,1,2)$. Note that the node labeled "8" in the splitting tree corresponds to the four vertexes at the first stage of the precedence graph; the two nodes labeled "4" correspond to the upper two vertexes and the lower two vertexes of the second stage of the precedence graph, and so on. The leaf nodes do not correspond to any vertexes in the precedence graph, but they are the outputs of the childless vertexes in the precedence graphs (e.g., in Fig. A.1(a), the two leaf nodes labeled with "2" are the outputs of vertexes $v_{1,0}$ and $v_{1,1}$). Thus, the number of stages in the splitting tree is one more than the number of stages in the precedence graph.

We could identify the isomorphism of two FFT precedence graphs by analyzing their splitting trees. If one splitting tree can be transformed to the other splitting tree by switching the positions of the two subtrees below a node, then the two FFT precedence graphs are isomorphic. Switching the positions of the two subtrees in the splitting tree as above corresponds to relabeling of a subset of vertexes of the precedence graph such that isomorphism applies. In Fig. A.1(a), for example, by switching the positions of the two subtrees below the root node "8", we get the splitting tree in Fig. A.1(b). The precedence graph of Fig. A.1(a) becomes the precedence graph of Fig. A.1(b) with the following vertex relabeling: $v_{1,0} \to v_{1,2}$, $v_{1,1} \to v_{1,3}$, $v_{1,2} \to v_{1,0}$, $v_{1,3} \to v_{1,1}$, $v_{2,2} \to v_{2,0}$, $v_{2,3} \to v_{2,1}$.

Note that isomorphism is a transitive relationship: if A is isomorphic to B because by switching the positions of two subtrees below a node, the splitting tree of A can be transformed to that of B, and B is isomorphic to C because by switching the positions of two subtrees below another node, the splitting tree of B can be transformed to that of C, then A and C are isomorphic. For example, in Fig. A.1(c), if we switch the positions of the two subtrees under the right node “4”, we get the splitting tree in Fig. A.1(d), and if we further switch the positions of the two subtrees under the left node “4”, we get the splitting tree in Fig. A.1(e). From Fig. A.1(e), if we switch the positions of the two subtrees under the right node “4”, we get the splitting tree in Fig. A.1(f). Thus, Fig. A.1(c), (d), (e), and (f) are isomorphic.

We can say that $(2, 2, 1, 1, 1, 1)$ and $(1, 1, 1, 1, 22)$ are isomorphic because their splitting trees are isomorphic. And we can also say that $(2, 1, 1, 2, 1, 1)$, $(2, 1, 1, 1, 1, 2)$, $(1, 1, 2, 1, 1, 2)$, and $(1, 1, 2, 2, 1, 1)$ are isomorphic for the same reason. The bin allocation $(2, 1, 1, 2, 1, 1)$, on the other hand, is not isomorphic to $(2, 2, 1, 1, 1, 1)$. Note, in particular, that although both $(2, 1, 1, 2, 1, 1)$ and $(2, 2, 1, 1, 1, 1)$ have two bins of size 2 and four bins of size 1, they are not isomorphic.

If the FFT precedence graphs G_1 and G_2 isomorphic, then if a scheduling algorithm can find the optimal schedule for G_1, then the algorithm should be able to find the optimal schedule for G_2. In fact, the performance of the scheduling algorithm does not change given the proper vertex relabeling. Thus, in studying the general performance of scheduling algorithms, we propose to focus on a set of non-isomorphic precedence graphs only so that isomorphic graphs are not over-represented if there are certain FFT graph structures with many isomorphic instances.

We denote the complete set of non-isomorphic FFT precedence graphs by F_n, and we denote the cardinality of F_n by f_n. We refer to the bin allocations that lead to isomorphic FFT precedence graphs as isomorphic bin allocations. Among isomorphic bin allocations, we only select one for our experiments, as they make no difference in the optimality of a scheduling algorithm. We refer to the selected bin allocation as the non-isomorphic bin-allocation instance (non-isomorphic instance in short). We denote the complete set of non-isomorphic bin-allocation instances for 2^n subcarriers by I_n. The elements in I_n map to the elements in F_n in a one-to-one manner, and the cardinality of I_n is also f_n.

Another question is how to select an instance from a group of isomorphic bin allocations for inclusion in I_n. We can randomly select any of the isomorphic bin allocations as it makes no difference in the optimality of a scheduling algorithm. However, we are interested in a deterministic test set. In this work, we select the "right heavy" bin allocation. In Fig. A.1, subfigure (c), (d), (d), and (f) shows the splitting trees for isomorphic bin allocations $(2, 1, 1, 2, 1, 1)$, $(2, 1, 1, 1, 1, 2)$, $(1, 1, 2, 1, 1, 2)$, and $(1, 1, 2, 2, 1, 1)$, respectively. In each splitting tree, there are two subtrees that split a node of "2" into two nodes of "1". We select $(2, 1, 1, 2, 1, 1)$ as the non-isomorphic bin allocation instance to include in I_n because there are more leaf nodes to the right in its splitting tree.

Appendix B

Recursion Function of f_n

This appendix derives f_n, the cardinality of F_n, or equivalently, the cardinality of I_n. Let $n = 2$, it is easy to verify that $f_2 = 4$. The following table list the elements in I_2.

Table B.1: Non-isomorphic instances for the fully packaged compact IFDMA system with four subcarriers

Index	Non-isomorphic Instance
0	$I_2(0) = (4)$
1	$I_2(1) = (2, 2)$
2	$I_2(2) = (2, 1, 1)$
3	$I_2(3) = (1, 1, 1, 1)$

We next explain how we obtain the non-isomorphic instances for $n = 3$, i.e., the elements in I_3. If there is only one IFDMA stream using up all bins, the corresponding non-isomorphic instance is (8). In the splitting tree, there is a root node "8" only. If there is more than one IFDMA stream, then for the splitting tree, there are two subtrees under the root node "8". Each subtree can be thought of as an instance in I_2. Thus, an instance in I_3, except (8), can be thought of as a combination of two instances in I_2. We can treat I_2 as occupying a dimension on a two-dimensional grid, as shown in Fig. B.1. Each combination is one point on this two-dimensional space. Fig. B.1 shows all possible combinations on the grid, where point (i, j) represents the combination of $I_2(i)$ and $I_2(j)$. Note that each combination in blue is isomorphic to a combination in red. Therefore, we only consider the combinations represented by points in red and black to remove isomorphism in I_3. In other words, we include only the points in the upper-left triangle of the grid.

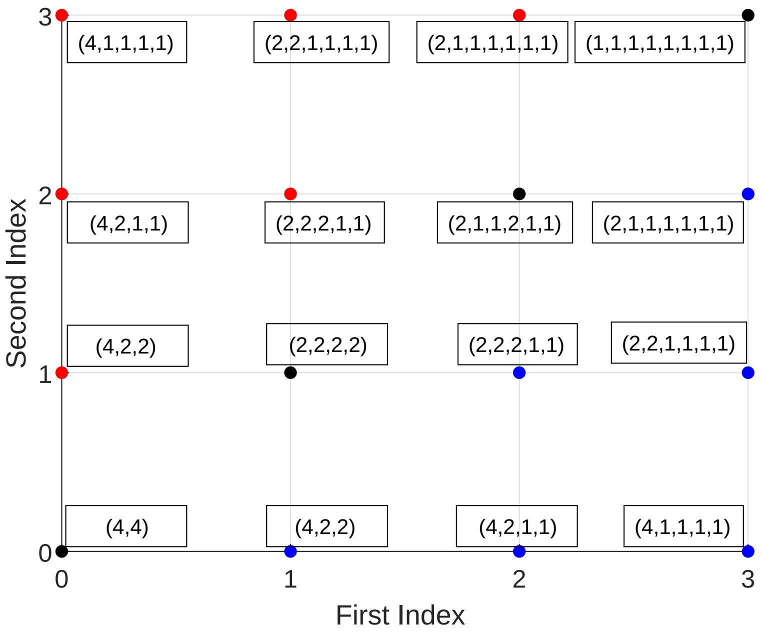


Figure B.1: An illustration of how non-single-stream non-isomorphic instances for eight-subcarrier IFDMA is generated.

Summing up the single-stream case and the non-single-stream cases, f_3 is then given by,

$$f_3 = 1 + (1 + 2 + ... + f_2) = 1 + \frac{f_2(f_2 + 1)}{2} = 11. \tag{B.1}$$

Table B.2 lists all the 11 non-isomorphic instances for $n = 3$.

The same procedure can be used to find I_{n+1} from I_n. Thus, we have

$$f_{n+1} = 1 + (1 + 2 + ... + f_n) = 1 + \frac{f_n(f_n + 1)}{2}. \tag{B.2}$$

In this way, we obtain f_n with various n. We list f_1 to f_{10} in Table B.3.

Table B.2: Non-isomorphic instances for the fully packaged compact IFDMA system with eight bins

Index	Non-isomorphic Instance	Index	Non-isomorphic Instance
0	$I_3(0) = (8)$	6	$I_3(6) = (2,2,2,1,1)$
1	$I_3(1) = (4,4)$	7	$I_3(7) = (2,2,1,1,1,1)$
2	$I_3(2) = (4,2,2)$	8	$I_3(8) = (2,1,1,2,1,1)$
3	$I_3(3) = (4,2,1,1)$	9	$I_3(9) = (2,1,1,1,1,1,1)$
4	$I_3(4) = (4,1,1,1,1)$	10	$I_3(10) = (1,1,1,1,1,1,1,1)$
5	$I_3(5) = (2,2,2,2)$		

Table B.3: Number of non-isomorphic instances for $n = 1, 2, ..., 10$

n	N	f_n	n	N	f_n
1	2	2	6	64	2598061
2	4	4	7	128	$3.3750E+12$
3	8	11	8	256	$5.6952E+24$
4	16	67	9	512	$1.6218E+49$
5	32	2279	10	1024	$1.3151E+98$

Appendix C

Test Set Generation Policy for $n \geq 7$

Note from Table B.3 that the number of non-isomorphic instances is not very large for $n \leq 6$. Therefore, when doing experiments to obtain $(\eta_{n,M}, \gamma_{n,M})$, we use I_n, the complete set of the non-isomorphic instance as the test set. Specifically, we can store all the instances in I_n in our program (e.g., as an array), and subject all the instances to the scheduling algorithm in a one-by-one manner in our experiments. The statistics $(\eta_{n,M}, \gamma_{n,M})$ can then be obtained from the experiments.

For $n \geq 7$, however, f_n is so large that it is impractical to test all the instances in I_n. Furthermore, even storing them in the program could be problematic. Therefore, we consider a subset of I_n as the test set and characterize $(\eta_{n,M}, \gamma_{n,M})$ statistically. We denote the subset by $I_n^{'}$, and the cardinality of $I_n^{'}$ is written as $f_n^{'}$. In our experiments, we set $f_n^{'} = \xi_0 = f_6$ (i.e., we fix the cardinality of $I_n^{'}$ to a constant $\xi_0 < f_n$) for all $n \geq 7$

Next, we explain how we obtain the non-single-stream elements $I_{n+1}^{'}$ with a recursive operation. We set the initial condition of the recursive analysis as $I_6^{'} = I_6$ and $f_6^{'} = f_6$. In this way, in the two-dimensional grid representing non-single-stream instances in the case of $n+1=7$, each dimension has the complete set I_6. We randomly select $f_n^{'} = \xi_0$ points from the upper left triangle of the two-dimensional grid $I_6 \times I_6$ plus the bin allocation (2^7) to form $I_7^{'}$. The f_7 instances are selected with equal probability to form the $f_7^{'}$

instances in $I_7^{'}$, the mechanism of which will be explained shortly.

For the two-dimensional grid $I_7 \times I_7$ that represents the $n+1=8$ case, I_7 is so large that it is impractical to have the complete set of I_7 in each dimension. Therefore, we use $I_7^{'}$, the random subset described above in place of I_7 in each of the two dimensions. We then randomly sample $f_n^{'} = \xi_0$ points from the upper left triangle of the $I_7^{'} \times I_7^{'}$ grid to form $I_8^{'}$. In the case of $n+1=9$, we randomly sample from the upper left triangle of the $I_8^{'} \times I_8^{'}$ grid to form $I_9^{'}$, and so on and so forth.

Next, we present how we perform the random selection of instances from the upper left triangle of $I_n^{'} \times I_n^{'}$ to form $I_{n+1}^{'}$, where $n+1 \geq 7$, to ensure that the instances are generated with equal probability.

First, we consider the instance where there is only one IFDMA stream, (2^{n+1}) . With a probability of $1/f_{n+1}$ (note that this is $1/f_{n+1}$ rather than $1/f_{n+1}^{'}$), we generate the non-isomorphic instance (2^{n+1}).

If the generation in the first step fails, we go to the non-single-stream instances and generate an element in $I_{n+1}^{'}$ by combining $I_n^{'}(i)$ and $I_n^{'}(j)$. In essence, we are trying to generate a duple (i,j) with constrains 1) $0 \leq i \leq f_n^{'} - 1$ and 2) $i \leq j \leq f_n^{'} - 1$ with equal probability. We denote the probability by a constant c, and we have

$$P_{i,j} = P_i \cdot P_{j|i} = P_i \cdot \frac{1}{f_n^{'} - i} = c, \tag{C.1}$$

where P_i is the probability of a specific i value is selected between 0 and $f_n^{'} - 1$. After i is selected, $P_{j|i}$ denotes the probability of a specific j is selected between i and $f_n^{'} - 1$.

We know that there are altogether $f_n^{'}\left(f_n^{'}+1\right)/2$ instances in the upper left triangle of $I_n^{'} \times I_n^{'}$ grid that form the non-single-stream elements in $I_{n+1}^{'}$. Normalizing the sum of the selection probabilities all instances to 1, we have

$$\sum_{i,j \geq i} P_{i,j} = \sum_{i,j \geq i} P_i \cdot P_{j|i} = \frac{f_n^{'}\left(f_n^{'}+1\right)}{2} c = 1. \tag{C.2}$$

Hence,

$$c = \frac{2}{f_n^{'}\left(f_n^{'}+1\right)}. \tag{C.3}$$

and P_i can be therefore written as,

$$P_i = c\left(f_n^{'} - i\right) = \frac{2\left(f_n^{'} - i\right)}{f_n^{'}\,(f_n^{'} + 1)}, i = 0, 1, ..., f_n^{'} - 1. \tag{C.4}$$

After an integer between 0 and $f_n^{'} - 1$ is assigned to i, with equal probability we select j between i and $f_n^{'} - 1$. We have

$$P_{j|i} = \frac{1}{f_n^{'} - i}, j = i, i + 1, ..., f_n^{'} - 1. \tag{C.5}$$

From the above equations, we can see that

$$\sum_{i=0}^{f_n^{'}-1} P_i = 1 \tag{C.6}$$

and

$$\sum_{i=0,...,f_n^{'}-1,\; j=i,...,f_n^{'}-1} P_{j|i} = 1 \tag{C.7}$$

are satisfied.

We now summarize how we generate (i, j) as follows:

1) We first generate an integer i between 0 and $f_n^{'} - 1$ probability $P_i = 2\left(f_n^{'} - i\right)/f_n^{'}\left(f_n^{'} + 1\right)$ (note: that means a larger integer has a smaller probability of being assigned to i).

2) We then generate an integer j between i and $f_n^{'} - 1$ with equal probability $P_{j|i} = 1/f_n^{'} - i$.

After we get (i, j), we can obtain an element of $I_{n+1}^{'}$ by combining $I_n^{'}(i)$ and $I_n^{'}(j)$.

Appendix D

Pseudo Code for MPS-FFT

Algorithm 1 MPS Algorithm ($H = 4$)

```
Input: IFDMA bin allocation $\{S_0, S_1, ...S_{R-1}\}$ and $M$
Output: Task schedule $\mathbf{X} = \{\chi_0, \chi_0, ...\chi_{T-1}\}$ and $T$
$t \leftarrow 0$                                              ▷ Algorithm initialization
$(\Omega,\ G(V, E),\ \langle P_1, P_2, P_3, P_4 \rangle) \leftarrow$ initialize func.
while $G(V, E) \neq \varnothing$ do
    $u \leftarrow 0$
    while ($u < M$ and $\Omega \neq \varnothing$) do                 ▷ Vertex selection
        $v_{i,j} \leftarrow \arg_{v \in \Omega} \max P\,(v)$
        $\chi_t\,.\,push\,(v_{i,j})$
        $\Omega\,.\,pop\,(v_{i,j})$
        if $v_{i,j}$ has companion $v_{i',j'}$ and $v_{i',j'} \in \Omega$ then    ▷ If $v_{i,j}$ is selected, raise $P_2$ of $v_{i',j'}$
            $P_2(v_{i',j'}) \leftarrow 2$
            $P(v_{i',j'}) \leftarrow$ priority scalar calculation func.
        end if
        $u \leftarrow u + 1$
    end while                              ▷ End of vertex selection in one time slot
    $V\,.\,pop\,(elements\,in\,\chi_t)$
    $E\,.\,pop\,(output\,edges\,of\,the\,elements\,in\,\chi_t)$
    $\Omega\,.\,push\,(new\,executable\,vertexes)$
    $t \leftarrow t + 1$
end while                                                   ▷ End of a time slot
```

Algorithm 2 Priority Scalar Calculation

Input: IFDMA bin allocation $\{S_0, S_1, ...S_{R-1}\}$

Output: Initialized (1)FFT precedence graph; (2)executable vertexes set Ω; (3)priority vectors $\langle P_1, P_2, P_3, P_4 \rangle$

```
$G(V,E) \leftarrow$ build FFT precedence graph with $\{S_0...S_{R-1}\}$   ▷ See Chapter 2.2.3 on graph building
for $i = log_2 N - 1$; $i \geq 0$; $i$-- do
    for $j = 0$; $j \leq log_2 N - 1$; $j$++ do
        if $v_{i,j} \in V$ then
            $K(v_{i,j}) \leftarrow$ child of $v_{i,j}$
            $k_{i,j} \leftarrow$ cardinality of $K(v_{i,j})$
            if $k_{i,j} = 0$ then
                $P_1(v_{i,j}) \leftarrow 0$
                $P_2(v_{i,j}) \leftarrow 0$
            else
                $P_1(v_{i,j}) \leftarrow 1 + \max_{v_{i+1,x} \in K(v_{i,j})} P_1(v_{i+1,x})$
                $P_2(v_{i,j}) \leftarrow 1$
            end if
            $P_3(v_{i,j}) \leftarrow k_{i,j}$
            $P_4(v_{i,j}) \leftarrow N/2 - 1 - j$
        end if
        if $i = 0$ then
            $\Omega.push(v_{i,j})$
        end if
    end for
end for
```

Algorithm 3 Priority Scalar Calculation

Input: Priority vector $\langle P_1(v_{i,j}), P_2(v_{i,j}), P_2(v_{i,j}), P_3(v_{i,j}) \rangle$

Output: Priority scalar $P(v_{i,j})$

```
$P(v_{i,j}) = \sum_{h=1}^{H=4} (N/2)^{H-h} P_h(v_{i,j})$   ▷ see eq. (2.15)
```

Appendix E

The Computation of IFDMA FFT is Computation Bounded

In this appendix, we compare the communication latency and computation latency within an FPGA chip to illustrate how the computation time of an FFT module dominates the overall processing latency.

Consider the XC7Z020 FPGA chip, a low-cost member of Xilinx's Zynq-7000 series that supports the AXI4 protocol for internal bus communication. The AXI4 bus can accommodate a data width of up to 1024 bits and can operate at frequencies as high as 400 MHz. In an ideal scenario, the theoretical bandwidth of the AXI4 bus is 1024 bits $\times$ 400 MHz $=$ 51.2 GB/s. In contrast, the output of a single butterfly computation is only eight bytes (two complex numbers outputs of a butterfly computation, with each consisting of a 16-bit real part and a 16-bit imaginary part). Under these ideal conditions, transmitting the output of one butterfly computation to the input of another butterfly computation in the next stage would take only 4 B $\div$ 51.2 GB/s $=$ 0.078 ns. In more practical terms, typical communication systems might use a 64-bit data width with a 250 MHz clock, resulting in a bandwidth of 16 Gbps (2 GB/s). This setup increases the transmission latency of a four-byte butterfly output to a more realistic 2 ns.

On the computational side, we examine an 8-point FFT implementation with a pipelined architecture, provided by Xilinx and synthesized on the XC7Z020 chip. This FFT implementation, which is realized with Vivado 2018.3, requires approximately 240 μs to complete a computation task. The pipelined 8-point FFT module has three processing elements, with the entire precedence graph containing twelve butterfly computations. Adopting a “top-down, left-to-right” scheduling policy, the pipelined module completes a task in nine time slots, yielding a hardware efficiency of $12 \div (3 \times 9) = 44.44\%$. Thus, each time slot – which includes both the inter-processor communication and intra-processor computation – takes $240\ \mu\text{s} \div 9 = 26.67\ \mu\text{s}$.

From this analysis, it is clear that the communication latency over the internal AXI4 bus is negligible compared to the duration of a single time slot (i.e., the combined communication and computation time). This finding supports our claim that FFT computation in compact IFDMA is computation-bound rather than communication-bound.

Appendix F

CFO's influence on autocorrelation

This appendix explains how CFO affects the autocorrelation of a general m-STS preamble sequence. The transmitted sequence is an m-STS preamble that follows:

$$s[n] = s\left[n+\eta\right] = ... = s\left[n+(m-1)\eta\right] \tag{F.1}$$

The received signal at the receiver is given as $y[n] = \sqrt{P}s[n] + w[n]$. A possible treatment of $y[n]$ for packet detections with the m-STS preamble is to divide the preamble sequence into $m/2$ pairs, with each pair containing two adjacent STSs. In this way, we use the average autocorrelation of these $m/2$ pairs of STSs as the packet-detection metric, which is[1]

$$a_m[n]=\frac{1}{m/2}\sum_{i=0}^{m/2-1} a[n+2i\eta]=\frac{2}{m\eta}\sum_{i=0}^{m/2-1}\sum_{k=0}^{\eta-1} y[n+2i\eta+k]y^*[n+(2i+1)\eta+k] \tag{F.2}$$

where $a[n]$ in the first line is given in (3.2).

To better isolate the effect of CFO, let us assume the noiseless case in this appendix (i.e., $w[n] = 0$). We start with the ideal case without CFO, for which we have

$$y[n]=\sqrt{P}s[n] \;\Leftrightarrow\; y[n]=y\left[n+\eta\right]=...=y\left[n+(m-1)\eta\right] \tag{F.3}$$

---

[1] We note that this may not be the only possible way to extend the treatment of a 2-STS preamble to a general m-STS preamble. A detailed introduction of possible treatments is given in [12]. Here we just choose a classic treatment to showcase the influence of CFO.

Substituting (F.3) to (F.2), we have the autocorrelation as

$$a_m[n] = \frac{2}{m\eta} \sum_{i=0}^{m/2-1} \sum_{k=0}^{\eta-1} |y[n+k]|^2 = \frac{P}{\eta} \sum_{k=0}^{\eta-1} |s[n+k]|^2 \tag{F.4}$$

We next take CFO into consideration. Let us first assume that the gap between XOs at the transmitter side and the receiver side is ϵ ppm. For example, if the measured frequency of the transmitter XO shifts from the rated value by +1 ppm, while that of the receiver XO shifts from the rated value by -4 ppm, then we have $\epsilon = +5$ ppm. Suppose that the frequency of the RF signal is f_{RF} Ghz, and the sampling period of the ADC at the receiver is T_S milliseconds. Due to CFO, sample n of the received signal is multiplied by $e^{jn\alpha}$, where $\alpha = 2\pi\epsilon T_S f_{RF}$ [91]. Therefore, given a m-STS preamble, for the substitution into (F.2), CFO dictates that

$$\begin{aligned} y[n+(2i+1)\eta] &= \sqrt{P} \cdot s[n+(2i+1)\eta] \cdot e^{j(n+(2i+1)\eta)\alpha} \\ &= y[n+2i\eta] \cdot e^{j\eta\alpha}, \ \ i = 0, ..., m/2-1 \end{aligned} \tag{F.5}$$

Substituting (F.5) into (F.2), we have

$$\begin{aligned} a_m[n] &= \tfrac{2}{m\eta} \sum_{i=0}^{m/2-1} \sum_{k=0}^{\eta-1} y[n+2i\eta+k] y^*[n+(2i+1)\eta+k] \\ &= e^{-j\eta\alpha} \cdot \tfrac{2}{m\eta} \sum_{i=0}^{m/2-1} \sum_{k=0}^{\eta-1} |y[n+k]|^2 \\ &= e^{-j\eta\alpha} \cdot \tfrac{P}{\eta} \sum_{k=0}^{\eta-1} |s[n+k]|^2 \end{aligned} \tag{F.6}$$

Comparing (F.4) and (F.6), we see CFO causes $a_m[n]$ to have a phase term of $e^{-j\eta\alpha}$. Without CFO, $a[n]$ is a real number. As CFO increases, part of the signal in $a[n]$ disperses to the imaginary part after the phase rotation, making $Re(a[n])$ smaller and $Im(a[n])$ larger.

We next give a concrete example of the practical insignificance of CFO. We assume a 2.4GHz WiFi system with a XO shift of $\epsilon = +5$ ppm (which is the worst case as discussed in Fig. 3.7) and 20MHz bandwidth, we have $T_S = 1/20MHz = 5 \cdot 10^{-5}$ ms. This gives $\alpha = 2\pi \cdot 0.0006$, which means the rotation per sample is $0.0006 \cdot 360 = 0.216$ degrees. If we further have $\eta = 16$ as defined in the IEEE 802.11 standard [65], the phase rotation in $a_m[n]$ is $\eta\alpha = 16 \cdot 0.216 = 3.456$ degrees and $e^{-j\eta\alpha} = 0.9982 - 0.0603i$, which is a relatively insignificant phase shift as far as communications are concerned.

Appendix G

The Optimal packet-detection threshold

This appendix presents a general methodology for selecting the packet-detection threshold ρ. To begin with, we formulate the general performance metric of our RP-S&C scheme as,

$$P_{gen} = \alpha_{FA} P_{FA} + \alpha_{MD} P_{MD} \tag{G.1}$$

where P_{FA} and P_{MD} denote the false-alarm and missed-detection probability given in (3.31) and (3.30), respectively. Further, α_{FA} and α_{MD} are the weights of P_{FA} and P_{MD}, respectively. Without loss of generality, we assume that

$$\alpha_{FA} + \alpha_{MD} = 1, \quad \alpha_{FA} > 0, \quad \alpha_{MD} > 0 \tag{G.2}$$

Apart from the constraint in (G.2), we also consider the practical constraint we put forth in Chapter 3.4.2, which is

$$P_{FA} \leq P_{FA}^{\max} \quad \text{and} \quad P_{MD} \leq P_{MD}^{\max} \tag{G.3}$$

where P_{FA}^{max} and P_{MD}^{max} are the maximum false-alarm and missed-detection probability allowed in the studied CSMA system. In Chapter 3.4.2 of the manuscript, we assume 10^{-6} and 10^{-4} for P_{FA}^{max} and P_{MD}^{max}, respectively. In essence, we are making a trade-off between "lower P_{FA}" and "lower P_{MD}" by adjusting the setting of ρ. In (G.1), the setting of α_{FA} and α_{MD} represents the relative importance of "lower P_{FA}" versus "lower P_{MD}".

We next explain how to find the packet detection threshold ρ given the target function in (G.1), the Pareto curve of the selected packet-detection scheme, and constraints of the optimization problem given in (G.2) and (G.3).

With the definition of P_{gen} in (G.1), we see that the original multi-objective optimization problem (i.e., lower P_{FA} and lower P_{MD}) is equivalent to the minimization of P_{gen}, which is an easier single-objective optimization problem. To visualize how we find the optimal P_{gen}, we rewrite (G.1) as

$$P_{MD} = -\frac{\alpha_{FA}}{\alpha_{MD}} P_{FA} + \frac{1}{\alpha_{MD}} P_{gen} \tag{G.4}$$

We can interpret (G.1) as a straight line, where P_{FA} and P_{MD} denote the x-axis and y-axis, respectively, and the slope of the line is $-\alpha_{FA}/\alpha_{MD}$. In this regard, the optimization problem defined above now becomes a linear programming (LP) problem [92]. We explain the LP problem and solve it as follows. In Fig. G.1 below, we plot 1) an example Pareto curve of a studied packet detection scheme, and 2) straight lines obtained according to (G.4) with different P_{gen} applied.

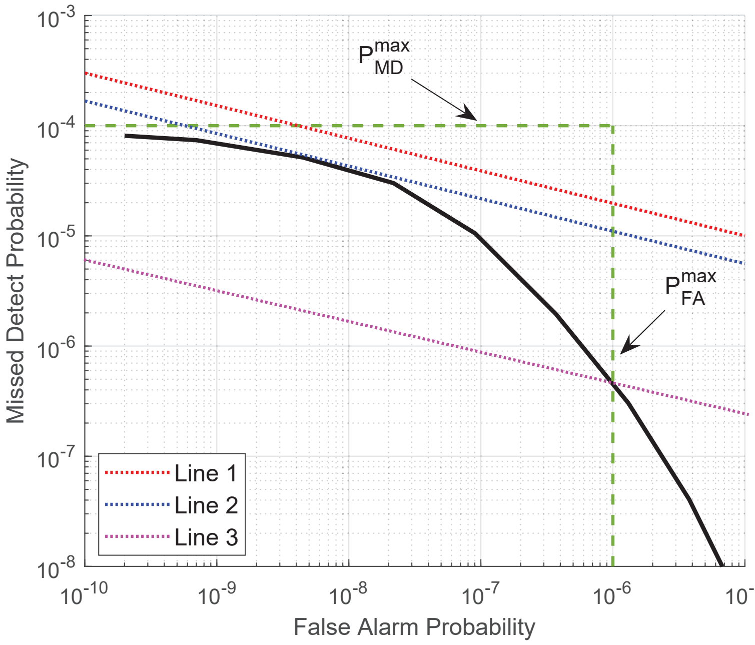


Figure G.1: How the LP problem could be solved.

Given the Pareto curve and these straight lines, we are interested in minimizing P_{gen}, which, according to (G.4), means we want a feasible line that has the smallest intercept. We note that P_{gen} is infeasible for the studied packet detection scheme if the corresponding straight line does not touch the Pareto curve (e.g., Line 1), as no valid point (P_{FA}, P_{MD}) exists. If we further lower the straight line by setting a smaller P_{gen}, we will have Line 2, the tangent of the Pareto curve. At this point, the desired P_{gen} becomes reachable.

We can further lower the straight line until we meet the boundary of the constraints (see (G.2) and (G.3) for the constraint). In Fig. G.1, the smallest P_{gen} is obtained when we have Line 3, which intersects the Pareto curve at the constraint boundary of $P_{FA} \leq P_{FA}^{max} = 10^{-6}$.

Appendix H

Packet Detections in Slow Fading Environments

This appendix investigates the packet detection process of an individual packet in the most common fading scenario in practice [93]. In slow fading, which is also known as block fading, the channel remains constant throughout the duration of a packet, although it may vary from packet to packet. We denote the constant phase shift of a packet by θ. For analysis purposes, we rewrite $y[n]$ as $y[n] = \sqrt{P}s[n]e^{j\theta}$.

The above formulation focuses on the fading phase only. For detailed analysis about the influence of CFO and channel noise, see our discussions in Chapter 3.2.4 and Appendix F for more details. With $y[n]$ we have

$$\begin{aligned} a[n] &= \frac{1}{\eta}\sum_{k=0}^{\eta-1}\left(\sqrt{P}s[n+k]e^{j\theta}\right)\left(\sqrt{P}s^*[n+\eta+k]e^{-j\theta}\right) \\ &= \frac{1}{\eta}\sum_{k=0}^{\eta-1} P\cdot s[n+k]\cdot s^*[n+\eta+k] \end{aligned} \tag{H.1}$$

$$\begin{aligned} b[n] &= \frac{1}{\eta}\sum_{k=0}^{\eta-1}\left(\sqrt{P}s[n+k]e^{j\theta}\right)\left(\sqrt{P}s^*[n+k]\cdot e^{-j\theta}\right) \\ &= \frac{1}{\eta}\sum_{k=0}^{\eta-1} P\cdot s[n+k]\cdot s^*[n+k] \end{aligned} \tag{H.2}$$

We see from the above that, as far as the detection of an individual packet is considered, the constant phase caused by slow fading has no influence on the calculation of $a[n]$ and $b[n]$ for that packet, as $e^{j\theta}$ in $y[n+k]$ and $e^{-j\theta}$ in $y^*[n+\eta+k]$ or $y^*[n+k]$ cancel each other.

Appendix I

Pareto Comparison versus Classification

We first formulate the binary classification treatment as follows. Suppose that we collect a dataset $\mathbf{X}$, which contains two types of samples: 1) Packet present 2) Packet absent. Given a testing sample, a packet detection scheme may claim packet detection or no packet detection, dividing the dataset into four types: 1) there is a packet, and the scheme successfully detects it; 2) there is a packet, but the scheme misses its detection; 3) there are no packets, but the scheme falsely claims that there is a packet; 4) there are no packets, and the scheme correctly deduces that there is no packet. For simplicity, we use α to represent the existence of a packet, i.e., $\alpha = 0$ for no packet and $\alpha = 1$ for the existence of a packet. And we use β to denote whether the scheme claims the detection of a packet, i.e., $\beta = 1$ and $\beta = 0$ for the receiver claiming the detection of a packet or not.

In the above analysis, Cases #2 and #3 are undesirable. The performance measure for this binary classification treatment can be a weighted sum of their probabilities, which is,

$$\begin{aligned} M &= a \cdot Pr\{\alpha = 1 \text{ and } \beta = 0\} + b \cdot Pr\{\alpha = 0 \text{ and } \beta = 1\} \\ &= a \cdot Pr\{\alpha = 1\} \cdot Pr\{\beta = 0 \mid \alpha = 1\} + b \cdot Pr\{\alpha = 0\} \cdot Pr\{\beta = 1 \mid \alpha = 0\} \end{aligned} \tag{I.1}$$

Given an arbitrary fixed dataset, $Pr\{\alpha = 1\}$ and $Pr\{\alpha = 0\}$ are constant values. Therefore, we can write M as,

$$M = c \cdot Pr\{\beta = 0 \mid \alpha = 1\} + d \cdot Pr\{\beta = 1 \mid \alpha = 0\} \tag{I.2}$$

where

$$c = a \cdot Pr\{\alpha = 1\} \text{ and } d = b \cdot Pr\{\alpha = 0\} \tag{I.3}$$

We explain below that if a packet detection scheme A is Pareto-superior to packet detection scheme B, then regardless of c and d, it is superior to scheme B in terms of the binary classification measure M, i.e., $M_A < M_B$.

Let us write the classification performance measures of schemes A and B as follows:

$$M_A = c \cdot Pr_A\{\beta = 0 \mid \alpha = 1\} + d \cdot Pr_A\{\beta = 1 \mid \alpha = 0\} \tag{I.4}$$

and

$$M_B = c \cdot Pr_B\{\beta = 0 \mid \alpha = 1\} + d \cdot Pr_B\{\beta = 1 \mid \alpha = 0\} \tag{I.5}$$

With (I.4) and (I.5), we have

$$\begin{aligned} M_A - M_B =& c \cdot (Pr_A\{\beta = 0 \mid \alpha = 1\} - Pr_B\{\beta = 0 \mid \alpha = 1\}) \\ & + d \cdot (Pr_A\{\beta = 1 \mid \alpha = 0\} - Pr_B\{\beta = 1 \mid \alpha = 0\}) \end{aligned} \tag{I.6}$$

The assumption of “scheme A is Pareto-superior to scheme B” is equivalent to the following statements:

Statement One: if schemes A and B can achieve the same missed-detection probability by adjusting their respective detection thresholds, then A has lower false-alarm probability, i.e., given $Pr_A\{\beta = 0 \mid \alpha = 1\} = Pr_B\{\beta = 0 \mid \alpha = 1\}$, we have $Pr_A\{\beta = 1 \mid \alpha = 0\} < Pr_B\{\beta = 1 \mid \alpha = 0\}$.

Statement Two: if schemes A and B can achieve the same false-alarm probability by adjusting their respective detection thresholds, then A has lower missed-detection probability, i.e., given $Pr_A\{\beta = 1 \mid \alpha = 0\} = Pr_B\{\beta = 1 \mid \alpha = 0\}$, we have $Pr_A\{\beta = 0 \mid \alpha = 1\} < Pr_B\{\beta = 0 \mid \alpha = 1\}$.

We prove $M_A - M_B < 0$ by substituting either Statement One or Statement Two into (I.5).

For the above analysis, we see that Pareto superiority is a stricter form of superiority. This superiority is independent of the testing dataset’s *a priori* probability on the existence (or absence) of a packet, i.e., specific values of $Pr\{\alpha = 1\}$ and $Pr\{\alpha = 0\}$.

Bibliography

[1] S. C. Liew and Y. Shao, "New transceiver designs for interleaved frequency-division multiple access," *IEEE Trans. Wireless Commun.*, vol. 19, no. 12, pp. 7765–7778, 2020.

[2] S. Han, Y. C. Liang, B. H. Soong, and S. Li, "Dynamic broadband spectrum refarming for OFDMA cellular systems," *IEEE Trans. Wireless Commun.*, vol. 15, no. 9, pp. 6203–6214, 2016.

[3] H. G. Myung, J. Lim, and D. J. Goodman, "Single carrier FDMA for uplink wireless transmission," *IEEE Veh. Technol. Mag.*, vol. 1, no. 3, pp. 30–38, 2006.

[4] Y. Zhu and K. B. Letaief, "CFO estimation and compensation in SC-IFDMA systems," *IEEE Trans. Wireless Commun.*, vol. 9, no. 10, pp. 3200–3213, 2010.

[5] Y. Du, J. Chen, Y. Lei, and X. Hao, "Performance analysis of nonlinear spatial modulation multiple-input multiple-output systems," *Digital Signal Process.*, vol. 115, p. 103064, 2021.

[6] S. Zhang, S. Xu, G. Y. Li, and E. Ayanoglu, "First 20 years of green radios," *IEEE Trans. Green Commun. and Netw.*, vol. 4, no. 1, pp. 1–15, 2020.

[7] Y. Shao and S. C. Liew, "Flexible subcarrier allocation for interleaved frequency division multiple access," *IEEE Trans. Wireless Commun.*, vol. 19, no. 11, pp. 7139–7152, 2020.

[8] C. Li, X. Wang, L. Cai, and Z. Gu, "Heterogeneous routing: A methodology for PPA tradeoff in network-on-chips," in *2018 IEEE 3rd International Conference on Integrated Circuits and Microsystems*, pp. 357–361, IEEE, 2018.

[9] H. Pan, T.-T. Chan, J. Li, and V. C. M. Leung, "Age of information with collision-resolution random access," *IEEE Trans. Veh. Technol.*, vol. 71, no. 10, pp. 11295–11300, 2022.

[10] Y. Feng, Z. Chen, Y. Jia, Z. Tian, L. Zhen, M. Wang, and D. O. Wu, "Irregular repetition slotted ALOHA scheme with short packets under rayleigh fading channels," *IEEE Trans. Veh. Technol.*, pp. 1–14, 2023.

[11] T. Schmidl and D. Cox, "Robust frequency and timing synchronization for OFDM," *IEEE Trans. Commun.*, vol. 45, no. 12, pp. 13–21, 1997.

[12] C. Liu, "On the design of OFDM signal detection algorithms for hardware relization," in *IEEE GLOBECOM*, vol. 2, pp. 596–599, 2003.

[13] P. Gupta and S. Ramaswamy, "Frame detection method for 802.11 b/g based WLAN systems," 2015. US Patent No.7733834.

[14] Mathworks, "Matlab WLAN toolbox documentation," 2023. [Online]. Available: `https://ww2.mathworks.cn/help/wlan`.

[15] Nuand, "BladeRF: source code and documentation," 2023. [Online]. Available: `https://github.com/Nuand/bladeRF`.

[16] Z. Xiang, S. Han, H. Peng, Y. Pei, and Y.-C. Liang, "A cross-layer analysis for symbiotic network using CSMA/CN protocol," *IEEE Internet Things J.*, vol. 8, no. 7, pp. 5697–5709, 2021.

[17] L. Iiyambo, G. Hancke, and A. M. Abu-Mahfouz, "A survey on NB-IoT random access: Approaches for uplink radio access network congestion management," *IEEE Access*, vol. 12, pp. 95487–95506, 2024.

[18] K. Song, B. Feng, Y. Wu, Z. Gao, and W. Zhang, "R-PMAC: A robust preamble-based MAC mechanism applied in industrial internet of things," *IEEE Internet Things J.*, vol. 11, no. 1, pp. 623–637, 2024.

[19] C. Shao, H. Park, and W. Lee, “DeWOZ: Rethinking the schmidl-cox algorithm for detecting Wi-Fi out of ZigBee interference,” *IEEE Wireless Commun. Lett.*, vol. 8, no. 1, pp. 289–292, 2019.

[20] M. L. Prasanna, “Synchronization of high speed OFDM links using variations of Schmidl Cox for SNR improvement,” in *2016 International Conference on Wireless Communications, Signal Processing and Networking*, pp. 279–283, 2016.

[21] J. Seo, J. Joo, G. Zhu, and S. C. Kim, “ESIM OFDM with schmidl and cox algorithm synchronizer in rayleigh fading channel,” in *2019 25th Asia-Pacific Conference on Communications (APCC)*, pp. 267–270, 2019.

[22] J. Cong, J. Lau, G. Liu, S. Neuendorffer, P. Pan, K. Vissers, and Z. Zhang, “FPGA HLS today: Successes, challenges, and opportunities,” *ACM Transactions on Reconfigurable Technology and Systems*, vol. 15, no. 4, pp. 1–42, 2022.

[23] B. Ahmad, S. Thakur, B. Tan, R. Karri, and H. Pearce, “Fixing hardware security bugs with large language models,” *arXiv preprint arXiv:2302.01215*, 2023.

[24] H. Pearce, B. Tan, and R. Karri, “Dave: Deriving automatically Verilog from English,” in *Proceedings of the 2020 ACM/IEEE Workshop on Machine Learning for CAD*, pp. 27–32, 2020.

[25] K. Chang, Y. Wang, H. Ren, M. Wang, S. Liang, Y. Han, H. Li, and X. Li, “ChipGPT: How far are we from natural language hardware design,” *arXiv preprint arXiv:2305.14019*, 2023.

[26] J. Blocklove, S. Garg, R. Karri, and H. Pearce, “Chip-Chat: Challenges and opportunities in conversational hardware design,” *arXiv preprint arXiv:2305.13243*, 2023.

[27] C. F. Hsiao, Y. Chen, and C. Y. Lee, “A generalized mixed-radix algorithm for memory-based FFT processors,” *IEEE Trans. Circuits Syst. II Express Briefs*, vol. 57, no. 1, pp. 26–30, 2010.

[28] M. Garrido, "A survey on pipelined FFT hardware architectures," *J. Signal Process. Syst.*, pp. 1–20, 2021.

[29] Y. T. Ma, "A VLSI-oriented parallel FFT algorithm," *IEEE Trans. Signal Process.*, vol. 44, no. 2, pp. 445–448, 1996.

[30] R. M. Karp, "Reducibility among combinatorial problems," in *Complex. Comput.*, pp. 85–103, Springer, 1972.

[31] J. K. Lenstra and A. Rinnooy Kan, "Complexity of scheduling under precedence constraints," *Oper. Res.*, vol. 26, no. 1, pp. 22–35, 1978.

[32] M. Bahtat, S. Belkouch, P. Elleaume, and P. Le Gall, "Instruction scheduling heuristic for an efficient FFT in VLIW processors with balanced resource usage," *EURASIP J. Adv. Signal Process.*, vol. 2016, no. 1, pp. 1–21, 2016.

[33] R. B. Ramakrishna, "Iterative modulo scheduling," 1995. [Online]. Available: `https://www.hpl.hp.com/techreports/94/HPL-94-115.pdf`.

[34] M. Frigo, "A fast fourier transform compiler," in *ACM PLDI*, pp. 169–180, 1999.

[35] M. Frigo and S. G. Johnson, "FFTW: An adaptive software architecture for the FFT," in *IEEE ICASSP*, vol. 3, pp. 1381–1384, IEEE, 1998.

[36] A. Ali and L. Johnsson, "UHFFT: A high performance DFT framework," 2006.

[37] M. Puschel, J. M. Moura, J. R. Johnson, D. Padua, M. M. Veloso, B. W. Singer, J. Xiong, F. Franchetti, A. Gacic, Y. Voronenko, *et al.*, "SPIRAL: Code generation for DSP transforms," *Proc. IEEE*, vol. 93, no. 2, pp. 232–275, 2005.

[38] D. Skinner, "Pruning the decimation in-time FFT algorithm," *IEEE Trans. on Acoustics, Speech, and Signal Process.*, vol. 24, no. 2, pp. 193–194, 1976.

[39] J. Markel, “FFT pruning,” *IEEE Trans. Audio Electroacoust.*, vol. 19, no. 4, pp. 305–311, 1971.

[40] T. Sreenivas and P. Rao, “Fft algorithm for both input and output pruning,” *IEEE Trans. on Acoustics, Speech, and Signal Process.*, vol. 27, no. 3, pp. 291–292, 1979.

[41] R. Airoldi, O. Anjum, F. Garzia, A. Wyglinski, and J. Nurmi, “Energy-efficient fast fourier transforms for cognitive radio systems,” *IEEE micro*, no. 6, pp. 66–76, 2010.

[42] C.-M. Chen, C.-C. Hung, and Y.-H. Huang, “An energy-efficient partial FFT processor for the OFDMA communication system,” *IEEE Trans. Circuits Syst. II Express Briefs*, vol. 57, no. 2, pp. 136–140, 2010.

[43] L. T. Melo, S. Santana, A. G. Silva-Filho, M. Lima, V. Medeiros, and M. L. Marinho, “An inter-FPGA communication bus with error detection and dynamic clock phase adjustment,” *Journal of the Brazilian Computer Society*, vol. 21, no. 1, p. 12, 2015.

[44] M. Abramowitz, I. A. Stegun, and R. H. Romer, “Handbook of mathematical functions with formulas, graphs, and mathematical tables,” 1988.

[45] R. Durrett, *Probability: theory and examples.* Cambridge university press, 2019.

[46] Xilinx, “Fast fourier transform v9.1 LogiCORE IP product guide,” 1995. [Online]. Available: `https://www.xilinx.com/support/documentation/ip_documentation/xfft/v9_1/pg109-xfft.pdf`.

[47] J. S. Kim, S. Lee, and M. Y. Chung, “Efficient random-access scheme for massive connectivity in 3GPP low-cost machine-type communications,” *IEEE Trans. Veh. Technol.*, vol. 66, no. 7, pp. 6280–6290, 2017.

[48] A. Gong, Y.-H. Lo, Y. Lin, and Y. Zhang, “Dynamic random access without observation under deadline-constrained periodic traffic,” *IEEE Trans. Veh. Technol.*, vol. 73, no. 1, pp. 1503–1508, 2024.

[49] J. Wu, W. Kim, and B. Shim, “Pilot-less one-shot sparse coding for short packet-based machine-type communications,” *IEEE Trans. Veh. Technol.*, vol. 69, no. 8, pp. 9117–9120, 2020.

[50] P. Li, X. Jian, F. Wang, S. Fu, and Z. Zhang, “Theoretical throughput analysis for massive random access with spatial successive decoding,” *IEEE Trans. Veh. Technol.*, vol. 69, no. 7, pp. 7998–8002, 2020.

[51] G. Maier, A. Paier, and C. F. Mecklenbräuker, “Packet detection and frequency synchronization with antenna diversity for IEEE 802.11 p based on real-world measurements,” in *2011 International Workshop on Smart Antennas*, pp. 1–7, 2011.

[52] B. Yin and X. Wei, “Efficient crowdsourced pareto-optimal queries over partial orders with quality guarantee,” *IEEE Trans. Emerg. Topics Comput.*, vol. 10, no. 1, pp. 297–311, 2022.

[53] R. Achenbach, M. Feuerstack-Raible, K. Meier, H. Rudolph, and R. Saur-Brosch, “A digitally temperature-compensated crystal oscillator,” *IEEE J. Solid-State Circuits*, vol. 35, no. 10, pp. 1502–1506, 2000.

[54] R. Fried and E. Rosin, “A high resolution frequency multiplier for clock signal generation,” *IEEE J. Solid-State Circuits*, vol. 31, no. 7, pp. 1059–1062, 1996.

[55] T.-H. Tran, H.-W. Peng, P. C.-P. Chao, and J.-W. Hsieh, “A low-ppm digitally controlled crystal oscillator compensated by a new 0.19-mm2 time-domain temperature sensor,” *IEEE Sens. J.*, vol. 17, no. 1, pp. 51–62, 2017.

[56] F. Zou, L. Zou, B. Yang, Q. Ma, X. Zou, J. Zou, S. Chen, D. Milosevic, Z. Cao, and H. Liu, “Optoelectronic oscillator for 5G wireless networks and beyond,” *Journal of Physics D: Applied Physics*, vol. 54, no. 42, p. 423002, 2021.

[57] J. Lin, “A low-phase-noise 0.004-ppm/step DCXO with guaranteed monotonicity in the 90-nm CMOS process,” *IEEE J. Solid-State Circuits*, vol. 40, no. 12, pp. 2726–2734, 2005.

[58] A. Miu, H. Balakrishnan, and C. E. Koksal, “Improving loss resilience with multi-radio diversity in wireless networks,” in *ACM Mobicom*, pp. 16–30, 2005.

[59] G. R. Woo, P. Kheradpour, D. Shen, and D. Katabi, “Beyond the bits: cooperative packet recovery using physical layer information,” in *ACM Mobicom*, pp. 147–158, 2007.

[60] M. Gowda, S. Sen, R. R. Choudhury, and S.-J. Lee, “Cooperative packet recovery in enterprise WLANs,” in *IEEE INFOCOM*, pp. 48–56, 2013.

[61] Y. Du, L. Hao, and Y. Lei, “SER optimization in OFDM-IM systems with nonlinear power amplifiers,” *IEEE Trans. Veh. Technol.*, vol. 72, no. 11, pp. 15046–15051, 2023.

[62] X. Huan, Y. Hao, K. Miao, H. He, and H. Hu, “Carrier frequency offset in IoT radio frequency fingerprint identification: An experimental review,” *IEEE Internet Things J.*, vol. 11, no. 5, pp. 59–73, 2024.

[63] M. Siniscalchi, F. Silveira, and C. Galup-Montoro, “Ultra-low-voltage CMOS crystal oscillators,” *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 67, no. 6, pp. 1846–1856, 2020.

[64] J. Everard, T. Burtichelov, and K. Ng, “Ultralow phase noise 10-MHz crystal oscillators,” *IEEE Trans. Ultrason., Ferroelectr., Freq. Control*, vol. 66, no. 1, pp. 181–191, 2019.

[65] I. W. Group, “IEEE standard for telecommunications and information exchange between systems,” *IEEE Std 802.11a-1999*, pp. 1–102, 1999.

[66] R. Durrett, *Probability: theory and examples.* Cambridge university press, 2019.

[67] M. Abramowitz, I. A. Stegun, and R. H. Romer, “Handbook of mathematical functions with formulas, graphs, and mathematical tables,” 1988.

[68] IEEE, “IEEE standard for information technology-local and metropolitan area networks-specific requirements,” *IEEE Std 802.11n-2009*, 2009.

[69] Y. Shao, E. Ozfatura, A. Perotti, B. Popovic, and D. Gündüz, "Attentioncode: Ultra-reliable feedback codes for short-packet communications," *IEEE Trans. Commun.*, pp. 1–1, 2023.

[70] M. Almekhlafi, M. A. Arfaoui, M. Elhattab, C. Assi, and A. Ghrayeb, "Joint scheduling of eMBB and URLLC services in RIS-aided downlink cellular networks," in *IEEE ICCCN*, pp. 1–9, 2021.

[71] R. Ali, Y. B. Zikria, A. K. Bashir, S. Garg, and H. S. Kim, "URLLC for 5G and beyond: Requirements, enabling incumbent technologies and network intelligence," *IEEE Access*, vol. 9, pp. 67064–67095, 2021.

[72] F. Euchner, M. Gauger, S. Dörner, and S. ten Brink, "A distributed massive MIMO channel sounder for big CSI data driven machine learning," in *25th International ITG Workshop on Smart Antennas*, 2021.

[73] M. Cummings and S. Haruyama, "FPGA in the software radio," *IEEE Commun. Mag.*, vol. 37, no. 2, pp. 108–112, 1999.

[74] X. Wang, J. Wei, D. Schuurmans, Q. Le, E. Chi, S. Narang, A. Chowdhery, and D. Zhou, "Self-consistency improves chain of thought reasoning in language models," *arXiv preprint arXiv:2203.11171*, 2022.

[75] Z. Zhang, A. Zhang, M. Li, and A. Smola, "Automatic chain of thought prompting in large language models," *arXiv preprint arXiv:2210.03493*, 2022.

[76] J. Wei, X. Wang, D. Schuurmans, M. Bosma, F. Xia, E. Chi, Q. V. Le, D. Zhou, *et al.*, "Chain-of-thought prompting elicits reasoning in large language models," *Advances in Neural Information Processing Systems*, vol. 35, pp. 24824–24837, 2022.

[77] H. Pearce, B. Ahmad, B. Tan, B. Dolan-Gavitt, and R. Karri, "Asleep at the keyboard? Assessing the security of Github copilot's code contributions," in *2022 IEEE Symposium on Security and Privacy (SP)*, pp. 754–768, IEEE, 2022.

[78] H. Pearce, B. Tan, B. Ahmad, R. Karri, and B. Dolan-Gavitt, "Examining zero-shot vulnerability repair with large language models," *arXiv preprint arXiv:2112.02125*, 2021.

[79] RapidSilicon, "Rapidgpt," 2023. [Online]. Available: `https://rapidsilicon.com/rapidgpt/`.

[80] S. Thakur, B. Ahmad, Z. Fan, H. Pearce, B. Tan, R. Karri, B. Dolan-Gavitt, and S. Garg, "Benchmarking large language models for automated verilog RTL code generation," in *2023 Design, Automation & Test in Europe Conference & Exhibition*, pp. 1–6, IEEE, 2023.

[81] Y. Du, S. C. Liew, and Y. Shao, "Efficient FFT computation in IFDMA transceivers," *IEEE Trans. Wirel. Commun.*, 2023.

[82] P. S. Stanimirović and M. D. Petković, "Gauss-jordan elimination method for computing outer inverses," *Applied Mathematics and Computation*, vol. 219, no. 9, pp. 4667–4679, 2013.

[83] J. J. Moré and D. C. Sorensen, "Newton's method," tech. rep., Argonne National Lab.(ANL), Argonne, IL (United States), 1982.

[84] S. Narang and A. Chowdhery, "Pathways language model (PaLM): Scaling to 540 billion parameters for breakthrough performance," *Google AI Blog*, 2022.

[85] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.

[86] Y. Chang, X. Wang, J. Wang, Y. Wu, L. Yang, K. Zhu, H. Chen, X. Yi, C. Wang, Y. Wang, *et al.*, "A survey on evaluation of large language models," *ACM Transactions on Intelligent Systems and Technology*, vol. 15, no. 3, pp. 1–45, 2024.

[87] H. Cui, Y. Du, Q. Yang, Y. Shao, and S. C. Liew, "Llmind: Orchestrating ai and iot with llm for complex task execution," *arXiv preprint arXiv:2312.09007*, 2023.

[88] K. Chen, Y. Du, T. You, M. Islam, Z. Guo, Y. Jin, G. Chen, and P.-A. Heng, “Llm-assisted multi-teacher continual learning for visual question answering in robotic surgery,” *arXiv preprint arXiv:2402.16664*, 2024.

[89] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, *et al.*, “Language models are few-shot learners,” *Advances in neural information processing systems*, vol. 33, pp. 1877–1901, 2020.

[90] K. Chen, H. Cao, J. Li, Y. Du, M. Guo, X. Zeng, L. Li, J. Qiu, P. A. Heng, and G. Chen, “An autonomous large language model agent for chemical literature data mining,” *arXiv preprint arXiv:2402.12993*, 2024.

[91] E. Sourour, H. El-Ghoroury, and D. McNeill, “Frequency offset estimation and correction in the IEEE 802.11a WLAN,” in *IIEEE VTC-Spring*, pp. 4923–4927, 2004.

[92] F. Zhang, Y. Du, K. Chen, Y. Shao, and S. C. Liew, “Addressing out-of-distribution challenges in image semantic communication systems with multi-modal large language models,” in *2024 IEEE WiOpt*, pp. 7–14, 2024.

[93] M. K. Simon and M.-S. Alouini, “Digital communications over fading channels (m.k. simon and m.s. alouini; 2005) [book review],” *IEEE Trans. Inf. Theory*, vol. 54, no. 7, pp. 3369–3370, 2008.