SPECTROMETER FRAMEWORKS FOR GRADIENT ARRAYS

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF ENGINEERING AND SCIENCE
OF BILKENT UNIVERSITY
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR
THE DEGREE OF
MASTER OF SCIENCE
IN
ELECTRICAL AND ELECTRONICS ENGINEERING

By
Mehmet Emin Öztürk
Sept 2025

Spectrometer Frameworks for Gradient Arrays
By Mehmet Emin Öztürk
Sept 2025

We certify that we have read this thesis and that in our opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

Ergin Atalar (Advisor)

Emine Ülkü Sarıtaş Çukur

Esin Öztürk Işık

Approved for the Graduate School of Engineering and Science:

Orhan Arıkan
Director of the Graduate School

ABSTRACT

SPECTROMETER FRAMEWORKS FOR GRADIENT ARRAYS

Mehmet Emin Öztürk
M.S. in Electrical and Electronics Engineering
Advisor: Ergin Atalar
Sept 2025

Gradient arrays are gaining popularity in the magnetic resonance imaging (MRI) community due to their superior performance and flexibility compared to conventional gradient coils. The technology can be used to excite multiple slices simultaneously, mitigate the $B1^+$ inhomogeneity, reduce the power loss on the cryostat, and increase peripheral nerve stimulation (PNS) thresholds. However, despite their potential, fully standalone MRI systems based on gradient arrays are yet to be realized. With an increasing number of channels, scalable design methodologies become essential to resolve the hardware complexity.

In this thesis, we present the development of modular spectrometer architectures for gradient array coils and parallel radio frequency (RF) transmit arrays to work towards a fully standalone gradient array magnetic resonance imaging system. Building upon previous work, the spectrometer submodules are implemented entirely on field programmable gate array (FPGA) hardware and consist of three main subsystems: a sequence parser, a gradient controller, and an RF controller.

The sequence parser reads PulSeq-formatted conventional sequences and converts them into hardware-compatible memory structures. The gradient controller accesses these memory-mapped sequences and generates time-resolved gradient amplitudes, which are projected onto array channels and sent to an external driver chain. Lastly, the RF controller synthesizes a carrier clock and drives external RF power amplifiers using pulse-width-modulated (PWM) envelope signals. Unlike traditional RF synthesizers that rely on digital-to-analog converters (DACs) or mixers, the RF clock is generated entirely using FPGA primitives, simplifying the overall architecture and reducing scaling costs.

Keywords: MRI spectrometer, Gradient array, RF synthesis.

ÖZET

GRADYAN DIZI SISTEMLERINE YÖNELIK MODÜLER SPEKTROMETRE MIMARILERI

Mehmet Emin Öztürk
Elektrik ve Elektronik Mühendisliği, Yüksek Lisans
Tez Danışmanı: Ergin Atalar
Eylül 2025

Gradyan sargı dizileri, geleneksel gradyan bobinlerine kıyasla sundukları üstün performans ve esneklik sayesinde manyetik rezonans görüntüleme (MRG) topluluğunda giderek daha fazla ilgi görmektedir. Bu teknoloji, tek bir RF darbesiyle aynı anda birden fazla dilimin uyarılmasını sağlayabilir, $B1^+$ alanındaki bozulmaları düzeltebilir, kriyostat üzerinde oluşan indüksiyon akımlarını indirgeyerek güç kaybını azaltabilir ve periferik sinir uyarımı (PNS) eşiklerini artırabilir. Ancak tüm bu potansiyele rağmen, çok kanallı gradyan dizi sistemlerini baz alan tam bağımsız MRG sistemleri henüz hayata geçirilememiştir. Kanal sayısının artmasıysa donanım karmaşıklığını artırmaktadır. Bu sorunu çözebilmek için de ölçeklenebilir tasarım metodolojileri zorunlu hâle gelmektedir.

Bu tez, tam bağımsız bir gradyan dizi MRG sistemi hedefi doğrultusunda, gradyan dizi bobinleri ve paralel radyo frekansı (RF) gönderim dizileri için modüler spektrometre mimarilerinin geliştirilmesini sunmaktadır. Önceki çalışmalara dayanarak, spektrometre alt modülleri tamamen FPGA sistemleri üzerinde uygulanmış ve üç alt sistem geliştirilmiştir: bir sekans çözücü, bir gradyan denetleyici ve bir RF denetleyici.

Sekans çözücü, PulSeq formatında hazırlanmış geleneksel sekansları okuyarak FPGA belleğiyle uyumlu bir yapıya dönüştürür. Gradyan denetleyici, belleğe alınmış bu sekansları kullanarak zamana bağlı gradyan genliklerini oluşturur; bu genlikler, dizi kanallarına yansıtılarak harici sürücü zincirine iletilir. Son olarak, RF denetleyici bir taşıyıcı frekans saat sinyali üretir ve darbe genişlik modülasyonu (DGM) ile şekillendirilmiş zarf sinyalleri kullanarak harici RF güç yükselteçlerini sürer. Geleneksel RF sentezleyicilerinin dijital-analog

dönüştürücüler (DAC) veya karıştırıcılar gibi bileşenlere olan gereksinimin aksine, bu sistemde RF saat sinyali yalnızca FPGA içindeki donanım kaynakları kullanılarak üretilmiştir. Bu yaklaşım, mimariyi sadeleştirmekte ve ölçekleme maliyetlerini azaltmaktadır.

Anahtar sözcükler: MRI spektrometresi, Gradyan dizisi, RF sentezi.

Preface

The research presented in this thesis has greatly benefited from the support, collaboration, and discussions I have had in the last three years.

Chapter 3 was developed in collaboration with colleagues from UMRAM. The sequence parser and gradient controller were designed with Fatma Gül Uyar. The experimental validation of the gradient controller was carried out together with Ege Aydın and M.Sc. Abdullah Erkam Arslan, the developer of the external gradient chain and the developer of previous RF power amplifiers and their corresponding controller architectures, respectively. The RF controller subsystem was also developed in collaboration with them, and the new design was adapted to a class-E RF power amplifier for parallel transmit applications. The contributions of Dr. Manouchehr Takrimi and Metin Can Işık in the gradient experiments also need mentioning.

The work in Chapter 5, in particular the waveform optimization algorithm in Section 5.1.1 and the export file format in Section 5.3 for gradient array sequences, was primarily carried out at the Magnetic Resonance Development and Application Center (MRDAC) of Uniklinik Freiburg. This was made possible through a summer research internship within the ERC project *MRITwins*, under the work package **Young Researcher Exchange**. During this period, I worked with Prof. Dr. Maxim Zaitsev and Dr. Sebastian Littin. Additional discussions with Dr. Qingping Chen, Mojtaba Shafiekhani, and Ariel Hannum also contributed to the development of this part. Lastly, Prof. Dr. Daniel B. Ennis from Stanford University introduced their work on GrOpt and provided valuable insights, which are referenced in this thesis.

Above all, my supervisor, Prof. Dr. Ergin Atalar, played a central role in shaping this thesis. He provided the research topics, ensured that the work was aligned with broader scientific goals, and guided the overall direction of the project from its conception to completion.

Finally, ChatGPT was used as a tool for refining sentences and improving the flow and delivery of the text.

Acknowledgement

First and foremost, I'd like to express my heartfelt gratitude to Prof. Ergin Atalar for his excellent guidance, kindness, and patience. There were many times I led astray, and each time he put me back onto the right track.

Then, I would like to thank the thesis committee, Assoc. Prof. Emine Ülkü Sarıtaş and Assoc. Prof. Esin Öztürk Işık for devoting their precious time and insightful comments.

I want to thank Dr. Manouchehr Takrimi for all his help throughout the years. It was always a pleasure to have those chats with him. I am also thankful to Dr. Reza Babaloo, Fatma Gül Uyar, Metin Can Işık, Dr. Sadeq AlSharafi, Dr. Mehdi Ghasemzadeh, and Muhammed Ali Khan for the time I spent with them. I relay my special thanks to Ege Aydın and Abdullah Erkam Arslan for being great colleagues and friends. Without them, half of this thesis would not be complete.

Apart from the group members, I thank Atakan Topçu, Hasan Atakan Bedel, Emirhan Koç, and Hasan Kayapınar for making UMRAM feel like home.

I especially thank my friends for always being there for me. Life would have been much harder and duller if not for them, especially Muhammed Melih Doğan.

Lastly, I thank my family who stressed over my stress, smiled for my happiness, and cried over my sadness for all their love and support. My mother and father, who are my unwavering pillars and guiding light, and my little siblings, who are my endless joy, I am ever grateful to you.

Special shout-out to Gökhan Kaplan, who provided endless supplies of delicious coffee and witnessed the first and last words of this thesis.

Contents

1 Introduction **1**

2 Preliminaries and Literature Review **3**

2.1 MRI Physics 3

2.1.1 Bloch Equations 3

2.1.2 RF Excitation 4

2.1.3 MR Signal 5

2.1.4 MRI Imaging Methods 5

2.1.5 K-space in MRI 6

2.1.6 Data Acquisition in MRI 8

2.2 Overview of Pulse Sequences in MRI 9

2.2.1 Designing Pulse Sequences 10

2.2.2 Oblique Imaging 11

2.2.3 PulSeq 16

2.3 Overview of MRI Spectrometers 20

2.3.1 Historical Context and Motivation for Homebuilt Systems 20

2.3.2 FPGA-Based Architectures and Real-Time Control 21

2.3.3 Multichannel Spectrometers and Scalable Architectures . . 22

2.3.4 Spectrometer Subsystems 23

2.4 Overview of Gradient Array Technology 29

2.4.1 Gradient Array Sequences 31

2.5 Chapter Summary 33

3 Spectrometer Architectures 35

3.1 Pulse Sequence Parsing and Data Preparation 36

3.1.1 Sequence Parser Subsystem 37

3.2 Gradient Controller Subsystem 42

3.2.1 Overview and Functional Flow 42

3.2.2 Memory Interface Submodule 43

3.2.3 Array Mapper Submodule 46

3.2.4 UART Submodule 47

3.3 RF Controller Subsystem 49

3.3.1 Overview and Functional Flow 50

3.3.2 RF Clock Generation 51

3.3.3 Other Submodules 59

3.4 Chapter Summary 60

4 Results of the Controller Subsystems 61

4.1 Gradient Controller Subsystem 61

4.2 RF Controller Subsystem 68

5 Pulse Sequence Design for Gradient Arrays 74

5.1 Designing Array Sequences on the Physical Coordinate System . . 76

5.1.1 Waveform Optimization Algorithm 77

5.1.2 Error-Free Rotation for Gradient Array Sequences 84

5.2 Dead-Period Minimization Algorithm 87

5.2.1 Results and Comparison of Dead Times with the Conventional Sequences 90

5.3 Export File Format for Array Sequences 92

5.4 Chapter Summary 96

6 Discussion and Conclusion 97

Appendix A 109

A.1 RF Submodules of the RF Controller Subsystem 109

A.1.1 RF Chain Submodule 109

A.1.2 Envelope PWM Submodule 111

A.1.3 Memory Interface Submodule 111

A.2 Legacy Feedforward Model of the Array Mapper Submodule . . . 112

Appendix B **114**

B.1 An Alternative Solution to the RF Synthesis 114

List of Figures

2.1 Visualization of the k-space trajectory under applied gradient fields 7

2.2 Allowable hardware limits, and the effect of oblique rotation. Reproduced from [25] 12

2.3 (a) An example non-oblique bSSFP pulse sequence designed for conventional three-channel MRI systems on logical coordinates. (b) The waveforms applied to the gradient amplifiers (physical coordinates) when the imaging plane is rotated by [x = 15°, y = 45°, z = 30°]. The segments that exceed the hardware limits, are indicated with bold lines. Reproduced from [26] 13

2.4 An example bSSFP GRE pulse sequence designed in the logical coordinate system, then regenerated in the physical domain after rotating by [x = 15°, y = 45°, z = 30°] 15

2.5 Visualization of the hierarchical dependence of different events . . 16

2.6 Visualization of the compression algorithm. Reproduced from [29] 20

2.7 Basic system architecture for a spectrometer 24

2.8 An example flow diagram for RF carrier synthesis 26

3.1 Overview of the combined gradient and RF controller subsystems. 35

3.2 An example sequence file segments (top row) and the resulting parsed file format with assigned address values (bottom row) . . . 39

3.3 Code snippet from an example header file. Different data types can be observed through different pointer definitions. The navigation from `[BLOCKS]` to `[RF]` to `[SHAPES]` can also be traced 40

3.4 Code snippet from an example TCL script 41

3.5 FPGA block design of the gradient controller subsystem 43

3.6 The internal structure of the Memory Interface submodule 44

3.7 Internal structure of the Array Mapper submodule 46

3.8 Internal structure of the UART submodule 48

3.9 FPGA block design of the RF controller subsystem 52

3.10 Internal structure of the RF Synthesis submodule 52

3.11 The output of the fractional clock divider generating a 123.238945 MHz clock signal (**o_pwm_0**), a reference clock signal (**clk_out_123MHz**) shown at the bottom, and the corresponding frequency response shown at the top. 58

4.1 Simulated waveforms from the Vitis HLS cosimulation platform (red lines) and the reference sequence plot 62

4.2 Time-averaged PWM waveforms for G_{ro} (green) and G_{pe} (yellow) channels and the reference sequence plot. Due to the low amplitude, G_{pe} appears as tiny ticks on the slopes of the G_{ro}. 63

4.3 Simulation results of the *Array Mapper* submodule. The array currents are projected from the G_{ss} 63

4.4 Experimental setup of the gradient driver chain. The flux gate sensors in the front are used as digital observers, while the ones on the back provide the feedback line over the RS32 cables back to the gradient driver board. 64

4.5 Current waveforms generated by the gradient pipeline. a) Simple trapezoidal waveforms with current weight matrix defined in Eq. 4.1, b) A more complex sequence with current weight matrix defined in Eq. 4.2. 65

4.6 Custom-built 40-channel gradient array coil used in the experimental setup. There are 12 primary and 12 shield coils with a total of 24 channels in the z-direction, and four primary and four shield coils with a total of 8 channels in each of the x and y-directions. . 66

4.7 Experimental setup, a) 4-channel full-bridge gradient amplifier, b) Programmed FPGA, c) Example sequence plotted on the FPGA, d) A custom-built gradient array coil inside the MRI to serve as the readout gradient. 67

4.8 Current outputs of separate gradient channels during the experiment. Maxwell pairs show almost identical inverted waveforms. The readout gradients' timing synchronization matches the rest of the sequence. 67

4.9 The resulting MR image of a tomato phantom obtained from the gradient pipeline 68

4.10 Comparison of the raw fractional clock divider output (blue) and `MMCM`-filtered output (green). 69

4.11 Observed lock time of the `MMCM` upon reactivation. 70

4.12 Evaluation of the filtered RF carrier waveform using an oscilloscope. 70

4.13 Observed delay for a 180° phase shift to take effect. 71

4.14 Example of a sinc-shaped RF envelope generated using PWM . . 72

4.15 MR images acquired during RF subsystem validation. a) RF controller operated continuously, revealing a zipper artifact. b) The RF controller is activated only during RF events, eliminating artifacts. 73

5.1 An example trapezoidal waveform 77

5.2 New quadrilateral waveform with adjustable peak corner points . 79

5.3 Binary search algorithm 80

5.4 An example 12-channel bSSFP GRE gradient array sequence designed in the logical coordinates and its rotated and regenerated counterpart in the physical coordinate system 86

5.5 Flow diagram explaining how to acquire the minimum dead-period duration 89

5.6 Flow diagram explaining how to generate the gradient array waveforms. The minimum dead period duration can be calculated by the process explained in Fig. 5.5 90

5.7 Comparison of the minimum possible D1, D2, TE, and TR values for conventional and array coils on five different oblique planes . . 91

5.8 Described Sequence Formatting - `[BLOCKS]` 94

5.9 Described Sequence Formatting - `[GRAD]` & `[TRAP]` 95

5.10 Described Sequence Formatting - `[SHAPES]` 95

A.1 Internal structure of the RF Chain submodule 110

A.2 Internal structure of the Envelope PWM submodule for the RF Controller subsystem 110

A.3 Internal structure of the Memory Interface submodule for the RF Controller subsystem 111

A.4 Simulation results of the *Array Mapper* submodule. The array currents are projected from the G_{ss} using the current weight matrix of [0.746, 0.1, -0.1, -0.746] to the corresponding voltage waveforms with channel parameters of: $R = 1\ \Omega$, $L = 150\ \mu H$. 113

B.1 Comparison of raw clock output and phase-adjusted clock output. The circles in the corner depict the exact time at which the RF clock must be generated to be at the Larmor frequency. 115

B.2 Frequency response of both the raw and phase-adjusted clock outputs 116

Chapter 1

Introduction

Conventional magnetic resonance imaging (MRI) systems rely on three large, orthogonal gradient coils aligned along the Cartesian axes to produce linear magnetic field gradients for spatial encoding. While this architecture has remained effective for decades, it ultimately has limited spatial flexibility and local encoding efficiency, particularly in the peripheral regions. These constraints are especially apparent in emerging high-throughput or high-performance systems, where stronger and faster gradient switching can induce unwanted physiological effects such as peripheral nerve stimulation (PNS) and simultaneously generate eddy currents that distort spatial encoding and degrade image quality.

To overcome the limitations of conventional gradient systems, multichannel coil architectures emerged as a promising alternative. By employing multiple, smaller, independently driven gradient coils, these systems reduced the load on gradient power amplifiers and enabled the generation of linear gradient fields with significantly greater flexibility. Early modular systems demonstrated the ability to improve the linearity of gradient fields, extend the field-of-view (FOV), and even enable multi-slice imaging capabilities [1–9]. These multi-coil systems—later termed matrix coils or gradient arrays—further offered the ability to dynamically synthesize linear and nonlinear spatial encoding fields [10], paving the way for anatomically adaptive, region-specific imaging protocols.

In more recent works, higher channel numbers were realized to further demonstrate the capabilities of the gradient array coils. Ertan et al. first demonstrated $B1^+$ inhomogeneity correction [11], then, through multi-slice excitation, obtained several images from different FOVs using a 9-channel z-coil system [12]. Later, Littin introduced an 84-channel arbitrarily positioned matrix coil system that enabled highly localized and non-bijective spatial encoding magnetic fields (SEMs) within a single imaging frame [13].

Despite their advantages and long history, multichannel gradient array systems remain confined to research settings, typically implemented as insert coils. A fully standalone gradient array system with a specialized pulse sequence development platform, integrated gradient, RF transmission and reception chains, and a central controller to establish communication and synchronization has yet to be realized.

This thesis builds upon the previous line of work [14–16] and introduces several subsystem frameworks for a fully standalone gradient array MRI: a sequence parser, a gradient controller, and an RF controller. The sequence parser reads PulSeq-formatted conventional sequences and converts them into hardware-compatible memory structures. The gradient controller accesses these memory-mapped sequences and generates time-resolved gradient amplitudes, which are projected onto array channels and sent to an external driver chain [15]. Lastly, the RF controller synthesizes a carrier clock and drives external RF power amplifiers using pulse-width-modulated (PWM) envelope signals. As opposed to traditional RF synthesizers that rely on digital-to-analog converters (DAC) or mixers, the RF clock is generated entirely using FPGA primitives, simplifying the overall architecture and reducing scaling costs. Together, the complete workflow from the design of pulse sequences to the synchronized control of gradient and RF subsystems is encompassed.

Additionally, a relatively unexplored field of pulse sequence design for gradient arrays is investigated. An error-free rotation algorithm for oblique imaging is presented. Lastly, an export file format for array sequences is proposed.

Chapter 2

Preliminaries and Literature Review

2.1 MRI Physics

Protons exhibit an intrinsic property called spin, which gives rise to a magnetic moment. When these spins are placed in a strong external magnetic field B_0, they begin to precess around the field direction. This precession leads to a small net magnetization vector that aligns with B_0 at a specific frequency, known as the *Larmor frequency* ω_0 [17], given by:

$$\omega_0 = -\gamma B_0 \tag{2.1}$$

2.1.1 Bloch Equations

Eq. 2.1, where γ is the gyromagnetic ratio, defines a fundamental relation of magnetic field B_0, and the corresponding Larmor frequency ω_0 in MRI. It forms the basis of the net magnetization vector of M_0, which can be decomposed into a

longitudinal component, M_z, and an orthogonal component, M_{xy}, also known as the transversal component. The magnetization vector can be phenomenologically described with the Bloch equation [18] shown in Eq. 2.2.

$$\frac{d\mathbf{M}}{dt} = \mathbf{M} \times \gamma\mathbf{B} - \frac{M_x\mathbf{i} + M_y\mathbf{j}}{T_2} - \frac{(M_z - M_0)\mathbf{k}}{T_1} \tag{2.2}$$

where $\mathbf{i}$, $\mathbf{j}$, and $\mathbf{k}$ are unit vectors in the x, y, and z direction, T_1 and T_2 are the relaxation terms characteristic of the object. The relaxation times T_1 and T_2 parameterize the decay of the magnetization back to the equilibrium after an excitation. The term T_1 describes how quickly the longitudinal component returns to equilibrium, while T_2 characterizes the decay of the transverse component. In cases where relaxation is ignored, the Bloch equations reduce to a simpler form:

$$\begin{pmatrix} M_x(t) \\ M_y(t) \\ M_z(t) \end{pmatrix} = \begin{pmatrix} \cos(\omega_0 t) & -\sin(\omega_0 t) & 0 \\ \sin(\omega_0 t) & \cos(\omega_0 t) & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} M_x(0) \\ M_y(0) \\ M_z(0) \end{pmatrix} \tag{2.3}$$

Since the object is placed in a static magnetic field B_0, the Bloch equation in Eq. 2.3 can be rewritten in a rotating coordinate system to simplify the analysis. The new coordinate system, called the *rotating frame*, rotates at the Larmor frequency about the direction of the main magnetic field. By reapplying the relaxation terms, the Bloch equation in the rotating frame can be found as in Eq. 2.4.

$$\left(\frac{d\mathbf{M}}{dt}\right)_{\mathbf{rot}} = -\frac{M_x\mathbf{i} + M_y\mathbf{j}}{T_2} - \frac{(M_z - M_0)\mathbf{k}}{T_1} \tag{2.4}$$

2.1.2 RF Excitation

An excitation pulse B_1 tuned to the Larmor frequency can tip the net magnetization from the z-axis. B_1 effectively applies a torque and disturbs the equilibrium.

This tipping, called the *flip angle*, results in the transversal component of M_{xy}. If B_1 is purely on the x-direction of the rotating frame, the net magnetization components in the rotating frame can be governed with Eq. 2.5.

$$\begin{aligned} M_{x,\mathrm{rot}} &= 0 \\ M_{y,\mathrm{rot}} &= M_z^0 \sin\left(\int_{\tau_0}^{\tau_{\mathrm{RF}}} \gamma B_{1,\mathrm{rot}}(t)\,\mathrm{d}t\right) \\ M_{z,\mathrm{rot}} &= M_z^0 \cos\left(\int_{\tau_0}^{\tau_{\mathrm{RF}}} \gamma B_{1,\mathrm{rot}}(t)\,\mathrm{d}t\right) \end{aligned} \tag{2.5}$$

2.1.3 MR Signal

Once the excitation pulse is applied, the magnetization vector is tipped into the transverse plane and begins to precess around the z-axis. According to Faraday's law of induction, this rotation produces measurable voltages in adjacent conducting objects, such as an RF-receive coil. The resulting transient signal can be described using a position-dependent magnetization density $m(x, y) = m_x(x, y) + im_y(x, y)$ as shown in Eq. 2.6.

$$s_r(t) = \int_x \int_y m(x, y) e^{-i\omega_0 t} dx dy \tag{2.6}$$

2.1.4 MRI Imaging Methods

Up to this point, the field was assumed to be a uniform magnetic field B_0. Apart from a non-selective excitation pulse, no other gradient fields were used to apply linear inhomogeneous fields. This led to uniform excitement of the object, with no apparent distinctions between different voxels.

To introduce spatial encoding, small linear magnetic field gradients—G_x and G_y—can be superimposed along the x and y axes, respectively. These gradients slightly perturb the local magnetic field strength across the object, resulting in

a spatially varying Larmor frequency. As a result, the signal acquired from the object becomes spatially encoded and is expressed as:

$$s_r(t) = \int_x \int_y m(x,y) e^{-i\omega_0 t} e^{-i\gamma G_x t} e^{-i\gamma G_y t} dx dy \tag{2.7}$$

When gradient fields G_x and G_y are applied across the transverse plane, the magnetic field becomes position-dependent. This variation causes the Larmor frequency to shift across space, so that each voxel precesses at a slightly different rate. As a result, voxels can be uniquely encoded based on their local frequency or phase, enabling spatial localization.

The same principle extends to the z-axis when a slice-selective gradient G_z is applied concurrently with an RF pulse B_1. In this case, G_z introduces a linear variation to the magnetic field strength along z, such that only spins within a specific slice have a Larmor frequency that matches the frequency of the RF excitation. Since excitation occurs only when the RF pulse is on-resonance with the local field, this method selectively excites a thin slice in the z-direction while leaving the rest of the volume unaffected.

After excitation, the transverse magnetization precesses and induces a time-varying voltage in the receive coil. By applying gradient fields during signal acquisition, each spin's phase evolves in a way that reflects its position in the object. As a result, spatial information becomes encoded into the signal itself. The measured signal $s_r(t)$ thus represents the superposition of all spatially modulated spin contributions. By continuously sampling this signal while sweeping the gradient fields, a trajectory is effectively traced in the k-space.

2.1.5 K-space in MRI

In magnetic resonance imaging, the raw data acquired during signal collection do not resemble an image in its original form. Instead, it exists within a domain called *k-space*. K-space is a Cartesian grid in the spatial frequency domain,

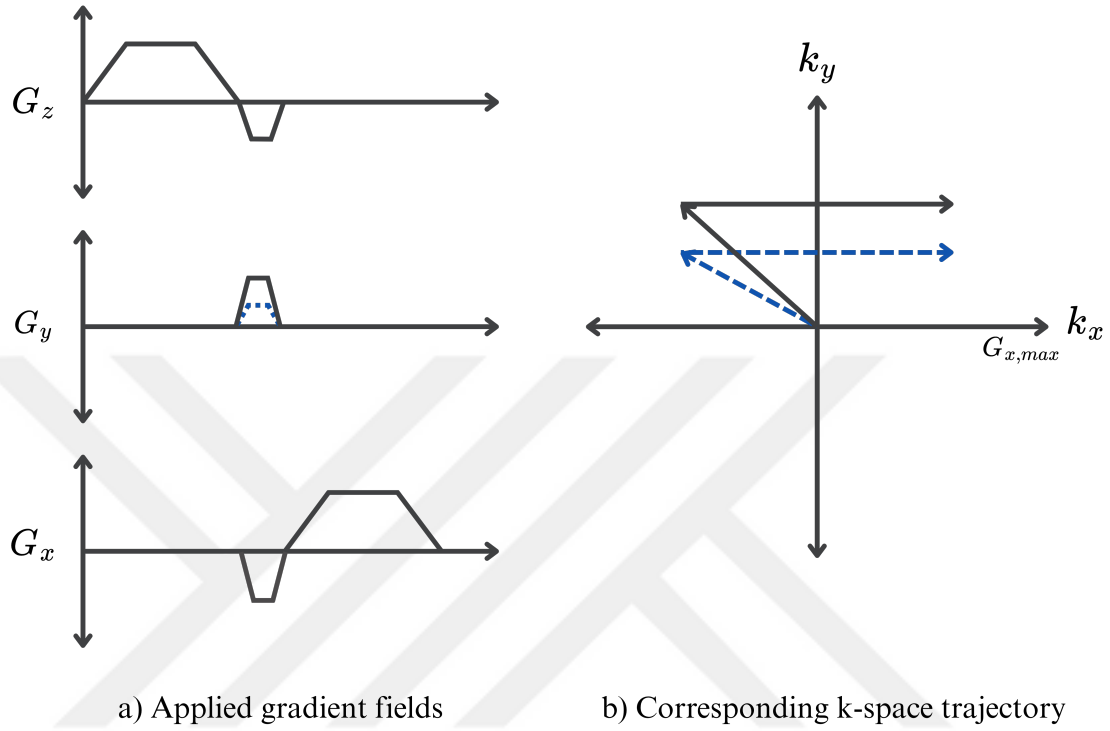


Figure 2.1: Visualization of the k-space trajectory under applied gradient fields

where each point contains information about the spatial variation of magnetization across the object being imaged.

The connection between k-space and spatial encoding arises from the effect of magnetic field gradients on the phase and frequency of precessing spins. As a gradient is applied, spins located at different spatial positions accumulate varying amounts of phase. The total phase accumulated by a spin at position $\mathbf{r} = (x, y, z)$ under a time-dependent gradient $\mathbf{G}(t)$ is given by:

$$\phi(t) = \gamma \int_0^t \mathbf{G}(\tau) \cdot \mathbf{r} \, d\tau \tag{2.8}$$

This phase accumulation serves as a form of spatial encoding through the Fourier transformation. To formalize this, the k-space trajectory $\mathbf{k}(t)$ is defined as:

$$\mathbf{k}(t) = \frac{\gamma}{2\pi} \int_0^t \mathbf{G}(\tau)\, d\tau \tag{2.9}$$

In this view, gradient waveforms act as navigators for the k-space trajectory. The accumulated area under each gradient lobe determines how far the trajectory moves in a particular direction. This behavior is illustrated in Fig. 2.1: a negative G_x and a positive G_y combination initially pushes the trajectory into the upper-left quadrant. Later, when a positive G_x is applied in isolation, the path continues horizontally with no vertical deviation. The blue and black traces reflect different G_y configurations, highlighting how gradient design dictates the sampling pattern in k-space.

2.1.6 Data Acquisition in MRI

Once the applied gradient waveforms define a specific k-space trajectory, the MR signal is continuously sampled as this trajectory unfolds. The signal induced in the receive coil reflects the net transverse magnetization across the object and carries spatially encoded phase information. Each sampled data point $s(t)$ corresponds to a single location $\mathbf{k}(t)$ in k-space.

$$s(t) = \int m(\mathbf{r})\, e^{-i2\pi \mathbf{k}(t)\cdot \mathbf{r}}\, d\mathbf{r} \tag{2.10}$$

As shown in Eq. 2.10, each acquired value represents a single spatial frequency component of the image. The full image is reconstructed after collecting a sufficiently dense set of these samples throughout k-space by performing an inverse Fourier transform.

In practice, signal acquisition occurs within a defined readout window—typically during the period when the readout gradient (e.g., G_x) is active and held constant. During this time, each ADC sample corresponds to a unique spatial frequency along the readout direction. To encode the orthogonal dimension, a separate gradient (e.g., G_y) is applied in discrete steps between successive

repetitions of the pulse sequence, which is commonly referred to as *phase encoding*.

The entire process is carefully synchronized with the gradient waveforms and RF excitation events. A complete imaging sequence, such as Spin Echo (SE) or Gradient Echo (GRE), incorporates prephasing and rephasing gradient lobes, slice selection, and a timed echo (TE) to ensure that the peak signal aligns with the center of k-space. The sampling bandwidth and ADC resolution directly influence the image resolution and field of view, and these parameters are selected according to the imaging requirements.

2.2 Overview of Pulse Sequences in MRI

Pulse sequences are at the heart of every MRI scan. They control when to send RF pulses, when to switch gradient fields, and when to collect data. By doing so, they steer the magnetization in a way that shapes the image contrast. At a basic level, a pulse sequence is just a timeline of instructions sent to the spectrometer. Each instruction tells the system exactly when to transmit RF energy, adjust gradients, or sample the signal.

Most pulse sequences begin with an RF excitation pulse, often paired with a slice-selective gradient to target a specific region within the imaging volume. This is followed by a combination of phase and frequency encoding gradients that spatially encode the signal through controlled variations in spin phase and frequency. Finally, the signal is sampled during the application of a readout gradient, and the data is stored in k-space for later reconstruction.

At a higher level, pulse sequences act as abstract descriptions of the imaging process. They are eventually translated into hardware-level commands with precise timing control. Key parameters, such as repetition time (TR), echo time (TE), flip angle, and gradient waveform shape, are all encoded within the sequence and play a direct role in determining image contrast, resolution, and scan

duration.

2.2.1 Designing Pulse Sequences

Designing a pulse sequence starts with selecting key imaging parameters, such as the FOV, slice thickness, flip angle, resolution, and scan timing. Other parameters include the orientation of the imaging plane, time-bandwidth product, number of readout and phase encoding steps, TE, TR, and receiver bandwidth. For accelerated scans, the parallel imaging factor also comes into play. These inputs define the shape of the k-space trajectory and determine the gradient areas that each segment of the sequence must cover.

The process usually begins by computing the area needed under each gradient lobe. For instance, the area under the readout gradient depends on the FOV and number of samples, while its amplitude and duration are adjusted based on the receiver bandwidth. The RF excitation pulse and slice-selective gradient are shaped according to the target flip angle, slice thickness, and slice profile. Event durations are then enforced to meet safety regulations, such as specific absorption rate (SAR) limits, which are particularly affected by short and intense RF pulses.

Since the RF and readout segments are tied directly to these imaging parameters, their timing and area requirements are typically fixed. They form the backbone of the pulse sequence and constrain how the surrounding gradients can be arranged.

In contrast, the periods between excitation and readout, also known as the dead periods, are more flexible. Although their start and end amplitudes are dictated by nearby events, and as long as the required area is preserved, the internal shape of these gradients can be freely optimized. These segments are often used for advanced features like gradient moment nulling, eddy current compensation, or reducing PNS.

Recent tools such as GrOpt [19] take advantage of this flexibility by adjusting

these gradient shapes to shorten the sequence or meet additional constraints. In Chapter 5.1.1, a similar approach is adopted to customize the dead period waveform shapes for gradient array applications

2.2.2 Oblique Imaging

Oblique imaging is widely used in clinical MRI to acquire images in planes that do not align with the scanner's standard axes. Instead of repositioning the patient, the gradient fields are rotated to match the orientation of anatomical structures to make the process faster and more comfortable [20–24].

A key example is cardiac MRI. The heart rarely lies along axial, sagittal, or coronal planes, and its long and short axes are typically oblique. To get meaningful views, the imaging plane must be rotated accordingly. However, the continuous beating of the heart and respiratory motion introduces additional complexity, making the task even more challenging. Even with techniques such as breath-hold cardiac imaging, scan durations must remain extremely short to minimize motion artifacts.

To meet these demands, various pulse sequences have been designed specifically for cardiac imaging [24]. Though successful, the ultimate limiting factor remained the system's hardware limits, such as the maximum gradient strength or, more importantly, the slew rate.

2.2.2.1 Logical-to-Physical Mapping

Pulse sequences are typically defined in a logical coordinate system, rather than in the physical magnet coordinates of the scanner (x, y, and z). The logical axes are named according to their functional roles during the sequence: slice selection (G_{ss}), phase encoding (G_{pe}), and readout (G_{ro}).

Pulse sequences are usually defined in an abstract coordinate system called the

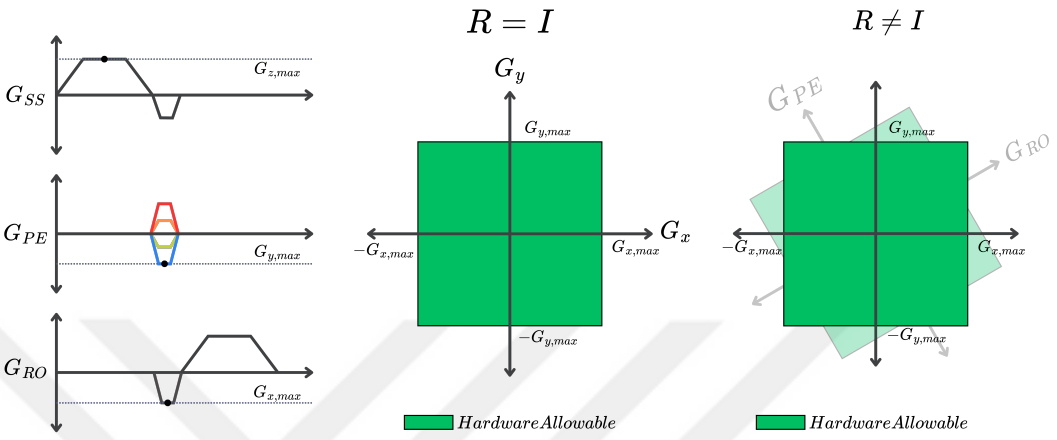


Figure 2.2: Allowable hardware limits, and the effect of oblique rotation. Reproduced from [25]

logical coordinate system rather than the scanner's physical magnet coordinates of (x, y, and z). The major axes of the logical coordinate system are named according to their functional roles during the sequence: slice selection (G_{ss}), phase encoding (G_{pe}), and readout (G_{ro}).

When designing the pulse sequence, a pure axial scan is usually assumed, where there are no rotations. Under this condition, gradient amplitudes and slopes remain within the operational limits of the hardware. The corresponding mapping onto a physical oblique plane is then applied using a rotation matrix R onto the three logical coordinates of G_{ss}, G_{pe}, G_{ro}. Thus, the resulting values of G_x, G_y, G_z, which denote the physical coordinate amplitudes to feed the hardware, can be found in Eq. 2.11.

$$\begin{bmatrix} G_x \\ G_y \\ G_z \end{bmatrix} = R \begin{bmatrix} G_{\mathrm{ro}} \\ G_{\mathrm{pe}} \\ G_{\mathrm{ss}} \end{bmatrix} \tag{2.11}$$

Assuming $G_{x,\mathrm{max}}$, $G_{y,\mathrm{max}}$, $G_{z,\mathrm{max}}$ are the maximum gradient strengths on each

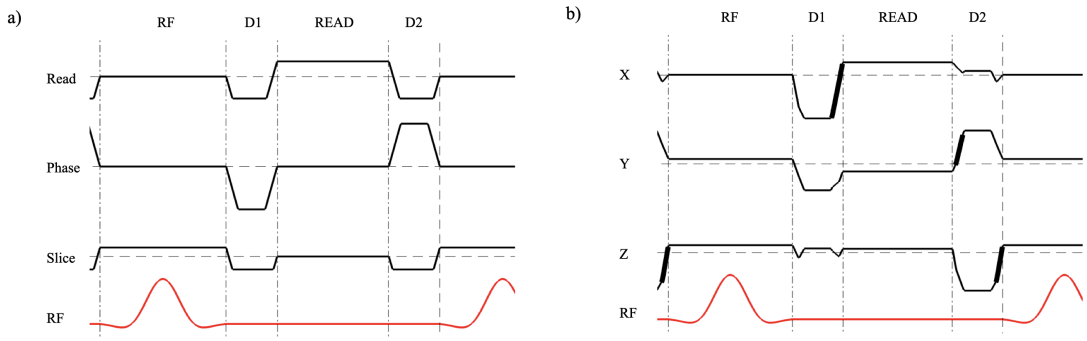


Figure 2.3: (a) An example non-oblique bSSFP pulse sequence designed for conventional three-channel MRI systems on logical coordinates. (b) The waveforms applied to the gradient amplifiers (physical coordinates) when the imaging plane is rotated by [x = 15°, y = 45°, z = 30°]. The segments that exceed the hardware limits, are indicated with bold lines. Reproduced from [26]

individual axis of (x, y, and z) on a conventional MRI system, R is the applied rotation matrix, and I is the identity matrix, we can see on Fig. 2.2 that, with $R = I$ the designed sequence is in the bounds of the hardware and can be synthesized without any problems. However, once the scan plane is rotated onto an oblique plane where $R \neq I$, it can be clearly seen that some of the previously available maximum gradient strength values become inaccessible in certain combinations.

The rotation operation not only limits the maximum available gradient strengths, but it also affects the maximum applicable slew rates. To further see the negative effects of the applied rotation matrix, an example balanced steady state free precession (bSSFP) GRE sequence can be designed, and a rotation matrix where $R \neq I$ can be applied, Fig. 2.3.

2.2.2.2 Hardware Optimized Trapezoids

One naive approach to ensure an error-free rotation is to limit the maximum limits imposed upon the logical coordinate system. As there are three major axes in a conventional MRI, decreasing the limits by $\sqrt{3}$ can guarantee a solution for every oblique plane. However, this approach is very conservative, and it can be detrimental, especially for demanding applications such as cardiac imaging.

Attempting to solve the hazards brought upon by oblique mapping, Bernstein and Licato [27] proposed an algorithm. Their works show that the rotated sequences do not always violate the hardware limits. The limitation must only be applied in specific combinations of gradient waveforms on specific oblique planes. Furthermore, their approach allows an on-the-fly calculation to relax the gradient limits, which is applicable to MR consoles. However, their algorithm is still conservative and limits the full potential of the hardware.

To optimally utilize the hardware on every rotation angle, Atalar and McVeigh [26] devised another algorithm. Their work proposed re-designing the pulse sequences on the physical coordinate system rather than the logical coordinate system. This way, the system limits need not be degraded, and the maximum gradient strength or the slew rate can be freely utilized in the sequence.

The main idea of their algorithm stems from the following statements [26]:

1. Trapezoidal waveforms with the highest slew rate ramps provide the shortest duration to cover a certain area.

2. Trapezoidal waveforms can be generated from three parameters: starting amplitude, ending amplitude, and area to cover.

3. The three parameters required to generate a trapezoidal waveform can be mapped to oblique planes using a rotation matrix Eq. 2.12.

$$\begin{bmatrix} s_x \\ s_y \\ s_z \end{bmatrix} = R \begin{bmatrix} s_{\mathrm{ro}} \\ s_{\mathrm{pe}} \\ s_{\mathrm{ss}} \end{bmatrix}, \begin{bmatrix} e_x \\ e_y \\ e_z \end{bmatrix} = R \begin{bmatrix} e_{\mathrm{ro}} \\ e_{\mathrm{pe}} \\ e_{\mathrm{ss}} \end{bmatrix}, \begin{bmatrix} a_x \\ a_y \\ a_z \end{bmatrix} = R \begin{bmatrix} a_{\mathrm{ro}} \\ a_{\mathrm{pe}} \\ a_{\mathrm{ss}} \end{bmatrix} \tag{2.12}$$

   where s and e represent the starting and ending amplitudes, and a represents the area to be covered by the waveform. The subscripts ro, pe, and ss represent the readout, phase encoding, and slice directions, and x, y, and z represent the directions in the physical coordinate system.

4. Using the mapped parameters, another trapezoidal waveform that uses the highest slew rate ramps can be generated, to obtain the shortest possible

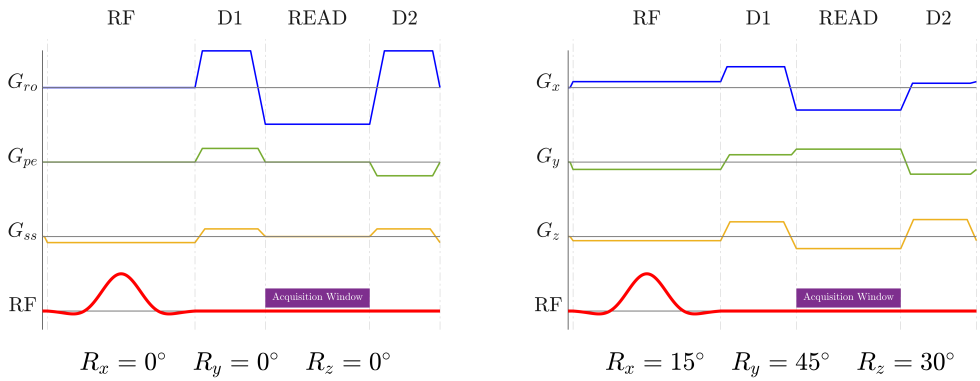


Figure 2.4: An example bSSFP GRE pulse sequence designed in the logical coordinate system, then regenerated in the physical domain after rotating by [x = 15°, y = 45°, z = 30°]

duration

As a result, hardware-optimized trapezoids can ensure the usage of the maximum hardware limits regardless of the imaging plane. Fig. 2.4 represents an example GRE sequence, and its counterpart on an oblique plane where the rotation angles are: $R_x = 15°, R_y = 45°, R_z = 30°$. When compared to the same operation shown in Fig. 2.3, the new waveform does not suffer from any hardware violation, while preserving the same TE and TR values.

2.2.2.3 Minimization of Dead Periods

Dead periods in pulse sequences are intervals where neither an RF nor a readout takes place. They are when the current magnetization is repositioned in the k-space. Thus, omitting the magnetic moments, the most critical parameter is the area that resides under the trapezoidal waveforms; their overall shapes do not matter. Leveraging this principle, Atalar and McVeigh designed an algorithm to minimize the dead times in the sequence using the hardware-optimized trapezoids. As a result, they obtain up to 30% and 40% reduction in the minimum possible TE and TR times, respectively [26].

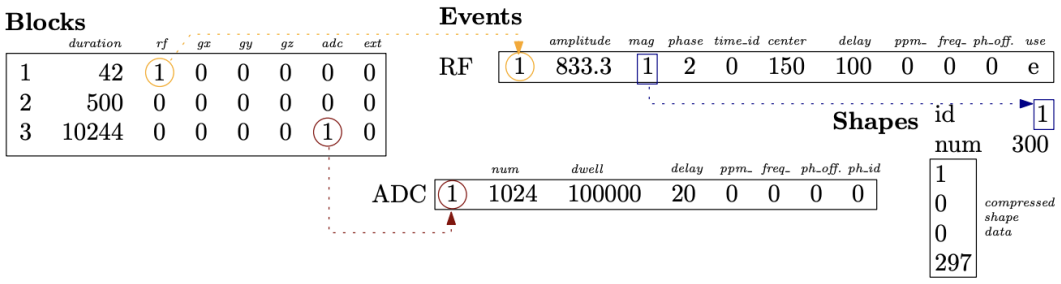


Figure 2.5: Visualization of the hierarchical dependence of different events

2.2.3 PulSeq

PulSeq [28] is a vendor-independent open-source magnetic resonance pulse sequence development framework developed by Zaitsev, Kroboth, and Layton at the University Medical Centre Freiburg. PulSeq provides a structured framework to define, test, and share MRI and NMR spectroscopy sequences across diverse hardware platforms without the constraints of vendor-specific software [29].

The toolbox emphasizes a human-readable format while ensuring precise timing control and compliance with specified hardware constraints. Sequences in PulSeq are structured into three hierarchical components: blocks, events, and shapes. At the highest level, each sequence block specifies a duration and indexes the concurrent events within that time window, such as RF pulses, gradient waveforms, or ADC sampling. These events, in turn, reference shape objects, which store the underlying waveform data, usually in a compressed format.

In the following subsections, some of the key events and functions will be explained in detail.

2.2.3.1 Events

PulSeq depends on a hierarchical structure based on its events. As seen in Fig. 2.5, the events are defined from a low-level object `[SHAPES]`, and rise till they reach the main description in the sequence under the high-level object of `[BLOCKS]`.

2.2.3.1.1 [BLOCKS] `[BLOCKS]` defines the flow of the sequence. Each block represents a specific time interval, which can consist of various low-level events. Each identification number in the `[BLOCKS]` corresponds to another event. It can be thought of as a navigation panel that leads to specific events that take place in the dedicated time interval.

2.2.3.1.2 [RF] [RF] events describe the RF excitation pulses that the hardware executes when the corresponding `[BLOCKS]` is reached. [RF] events contain several key information: ID, amplitude, shape ID, phase ID, time ID, center point, delay, phase offset, and frequency offset.

- **ID:** indicates the ID from the `[BLOCKS]`.
- **Amplitude:** scaling factor for the RF shape.
- **Shape ID:** indicates the ID of the designated RF shape.
- **Phase ID:** indicates the ID of the designated RF phase.
- **Time ID:** indicates the ID of the designated time vector.
- **Center Point:** indicates when the center of the RF pulse is reached.
- **Delay:** execution is delayed for the specified delay time.
- **Phase Offset:** indicates the required phase offset for the RF.
- **Frequency Offset:** indicates the required frequency offset for the RF.

2.2.3.1.3 Gradients Gradients fall under two different objects, `[GRAD]` and `[TRAP]`. The ID used in `[BLOCKS]` does not distinguish between these events and regards both as gradient objects. Therefore, it is possible to see non-consecutive numeration on these events.

2.2.3.1.3.1 [GRAD] Generalized gradient object. It consists of several information: ID, amplitude, shape ID, time ID, and delay. `[GRAD]` object can also be described under two different categories: extended gradients and arbitrary gradients. While both correspond to the same `[GRAD]` object, there is a subtle difference between the two. Arbitrary gradients define the full transient gradient waveform, whereas extended trapezoids only mention the specific time points. Therefore, extended trapezoids usually occupy less space.

- **ID:** indicates the ID from the `[BLOCKS]`.
- **Amplitude:** scaling factor for the gradient shape.
- **Shape ID:** indicates the ID of the designated gradient shape.
- **Time ID:** indicates the ID of the designated time vector. Points defined in the shape ID are executed on the corresponding time points. The shape is assumed to be linear between consecutive time points and is subject to linear interpolation.
- **Delay:** execution is delayed for the specified delay time.

2.2.3.1.3.2 [TRAP] Specialized gradient object. If the gradient object is a trapezoidal waveform that starts and ends on a zero amplitude gradient, then the definition is simplified without the necessity to address any shape objects. It consists of several information: ID, amplitude, rise time, flat time, fall time, and delay.

- **ID:** indicates the ID from the `[BLOCKS]`.
- **Amplitude:** maximum amplitude of the trapezoid.
- **Rise Time:** indicates the time it takes for the amplitude to rise from zero to the maximum amplitude declared by "Amplitude".
- **Flat Time:** indicates the time it takes for the amplitude to stay at the maximum amplitude.

- **Fall Time:** indicates the time it takes for the amplitude to fall from the maximum amplitude to the zero line.
- **Delay:** execution is delayed for the specified delay time.

2.2.3.1.4 [SHAPES] `[SHAPES]` generally stores a large amount of information, be it the shape of the waveforms, corresponding time points, or the transient phase of the RF pulse. `[RF]` and `[BLOCKS]` events actively access `[SHAPES]` as the bulk of the waveform information is stored in that object. It consists of an identification ID, the number of samples, and the data points. Sometimes the data stored inside is compressed to save space. Caution is required to identify compressed and uncompressed shapes as they are interpreted differently, especially when reading from a hardware perspective. If the shape is uncompressed, then each amplitude corresponds to a time point, defined in the time vector. However, if the shape is compressed, the values defined in the shape object represent the first derivative of a certain time interval. Therefore, the amplitude values are obtained by adding up the first derivative.

2.2.3.1.4.1 Compression Algorithm As different shapes can occupy a large amount of space, a compression algorithm can help reduce it significantly. The compression algorithm is applied if the shape values have a first derivative that repeats for more than four points. The first derivatives of the whole shape are found, and any repetitions are represented as integers. As a result, the compressed shape defines the first derivatives of the values and cannot be interpreted the same as the uncompressed data.

To identify if the shape is compressed or not, one can check the number of stored data inside the shape object and compare it with the number of samples defined at the beginning of the object. To uncompress the data, one needs to add the stored values and keep track of repeated numbers. If two consecutive data points are the same, then the next value is an integer indicating how many times the derivative must be added. A visualization of the compression algorithm can be found in Fig. 2.6.

Shape		Step 1 (derivative)		Step 2 (compression)
0.0		0.0		0
0.1		0.1		0.1
0.25		0.15		0.15
0.5		0.25		0.25
1.0		0.5		0.5
1.0		0.0		0.0
1.0	→	0.0	→	0.0
1.0		0.0		4
1.0		0.0		-0.25
1.0		0.0		-0.25
1.0		0.0		2
0.75		-0.25		
0.5		-0.25		
0.25		-0.25		
0.0		-0.25		

Figure 2.6: Visualization of the compression algorithm. Reproduced from [29]

2.3 Overview of MRI Spectrometers

The spectrometer is the central control unit of an MRI system. It governs the operational pipeline from pulse sequence interpretation to the coordination of hardware components. It interprets the sequence file and distributes the relevant information to mission-critical subsystems such as the gradient, RF, and receiver. Using dedicated trigger lines and control signals ensures the synchronized execution of parallel processes.

2.3.1 Historical Context and Motivation for Homebuilt Systems

Commercial MR spectrometers are designed to be robust and versatile, but they are not always suitable for research that requires hardware-level customization. Since MRI is an ever-developing field, many researchers were motivated to build their own spectrometer systems or subsystems to explore new experimental ideas [30].

The earliest homebuilt spectrometers were mostly analog. Researchers built separate modules for RF generation, gradient control, and reception, often using discrete components and analog timing circuits. These systems enabled basic waveform shaping and signal acquisition, but suffered from several limitations such as drift, sensitivity to noise, and difficulty in maintaining synchronization across channels [31]. Despite these challenges, they were useful for early MR microscopy, spectroscopy, and low-field imaging studies.

As digital electronics became more accessible, microprocessors and programmable logic started finding their ways into spectrometer design [31–36]. The shift to digital spectrometers brought significant improvements in signal quality, timing precision, and system modularity. Instead of relying on analog mixers and detectors, digital systems used numerically controlled oscillators (NCOs) for waveform synthesis and phase modulation. This eliminated carrier leakage and nonlinearity associated with analog signal chains [31–33]. On the receiver side, digital quadrature detection and filtering were implemented using finite impulse response (FIR) filters, which provided high stability and customizable frequency responses [34]. These features reduced the need for analog calibration and allowed for more compact, reprogrammable spectrometer designs.

2.3.2 FPGA-Based Architectures and Real-Time Control

The adoption of FPGA-based architectures marked a major turning point in spectrometer design. These platforms offered both reconfigurability and real-time control, making them ideal for managing RF pulses, gradient waveforms, and timing synchronization. One of the earliest examples was developed by Takeda, who implemented a digital broadband NMR spectrometer using an FPGA-based pulse programmer and phase accumulator to generate arbitrary waveforms and flexible sequence control [30].

Following this, the Medusa platform introduced a scalable architecture using multiple USB-connected processing units [37]. It supported synchronized multi-channel operation for both RF transmission and reception. Each unit contained

an FPGA and DSP for localized processing. Medusa was among the first open frameworks designed for parallel transmission and signal acquisition, and it laid the groundwork for distributed spectrometer design.

Later systems like OCRA built on this modular philosophy by offering a low-cost, open-source MRI console that could control gradient and RF hardware in real time using a centralized FPGA [38]. OCRA emphasized accessibility and real-time feedback, allowing researchers to directly prototype and test pulse sequences with minimal latency.

Most recently, MaRCoS extended these ideas to low-field MRI applications. As a fully open-source spectrometer framework, MaRCoS supported gradient and RF waveform control, digital triggering, and optional receiver support [39]. Its flexible and lightweight architecture made it suitable for benchtop and portable MRI systems.

2.3.3 Multichannel Spectrometers and Scalable Architectures

With the advancement of MRI technology, the demand for spectrometers capable of supporting multiple RF transmit and receive channels has grown significantly. This requirement is primarily driven by applications such as parallel transmission (pTx), parallel imaging, B_1 shimming, and multi-coil receive arrays. Early multi-channel architectures were developed to address this need, starting with scalable platforms like Medusa [37].

The Medusa system introduced a modular, USB-connected architecture where each unit controlled one or more RF channels using dedicated FPGA and DSP cores. This distributed design enabled both multichannel RF transmission and simultaneous receive chains, synchronized through USB timing and host control. Medusa demonstrated how separate modules could be used in parallel to form a scalable spectrometer architecture adaptable to a wide range of experimental setups.

Expanding on this idea, MaRCoS [39] was later extended to include up to four receive channels in its revised architecture [40]. The updated design employed multiple FPGA peripherals interfacing with a central controller, enabling real-time data streaming and per-channel configuration. The modular nature of the system made it well-suited for both low- and mid-field MRI platforms, particularly those with open-access geometries or unconventional hardware arrangements.

Later, the MaRCoS system [39] was revised to include support for up to four receive channels [40], using multiple FPGA-based subsystems linked to a central controller. This configuration supported real-time data streaming and per-channel configuration. Its modular nature made it compatible with both low- and mid-field MRI systems, including those with open or unconventional geometries.

In high-field imaging, further advances were demonstrated with developing a spectrometer for 14 T MRI [41]. This system incorporated synchronized multi-channel RF generation, allowing precise control over waveform parameters such as frequency and phase. Real-time feedback mechanisms enabled adaptive RF transmission, supporting advanced techniques required for ultra-high field applications.

Together, these systems highlight the shift from single-channel architectures to scalable, open, and multichannel spectrometer platforms. As modern MRI continues to demand increased flexibility and parallelization, multichannel spectrometer designs will remain a critical component of future systems.

2.3.4 Spectrometer Subsystems

A typical MRI spectrometer consists of several digital subsystems, including a pulse sequence interpreter, a gradient waveform generator, and an RF transmitter. Figure 2.7 illustrates the core structure of a basic spectrometer and the interconnections among its components.

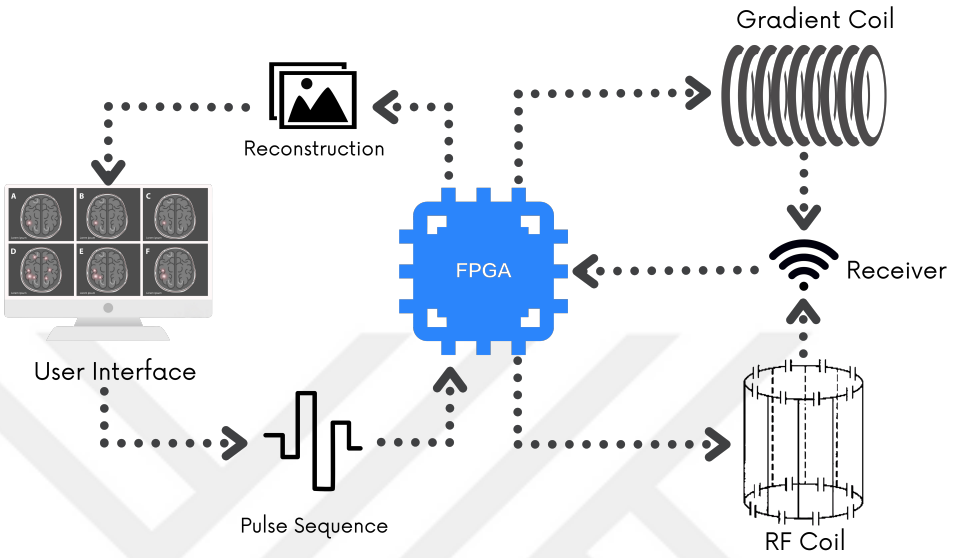


Figure 2.7: Basic system architecture for a spectrometer

2.3.4.1 Pulse Sequence Interpreter

The pulse sequence interpreter processes the sequence definition file and extracts information such as RF pulses, gradient events, and ADC readouts. In commercial systems, this component typically runs as software on a host computer, translating high-level sequence instructions into hardware-compatible commands. In contrast, research-focused and custom-built systems often integrate the interpreter directly into the FPGA fabric, reducing latency and allowing more deterministic control.

In FPGA-based designs, the sequence is parsed offline and stored in memory as a series of structured timing blocks. At runtime, a state machine or microcoded controller steps through these blocks, issuing synchronized instructions to the RF and gradient subsystems. This structure supports precise alignment of waveform timing, including phase control, programmed delays, and conditional execution. Some platforms, such as MaRCoS and OCRA, also allow reconfiguration between blocks, enabling dynamic or adaptive pulse sequences during acquisition.

2.3.4.2 Gradient Waveform Generator

The gradient waveform generator produces time-varying amplitudes that drive the gradient amplifiers and coils, defining the spatial encoding of the MR signal. These amplitudes must be delivered with high temporal resolution and accuracy, typically updated at a fixed raster interval ranging from 5 to 10 microseconds.

In FPGA-based spectrometers, gradient values are either stored as precomputed samples in memory or calculated in real time during each sequence block. A timing controller ensures that these samples are delivered at regular intervals and synchronized with the RF and ADC events.

Additional compensation mechanisms are often employed to ensure accurate delivery of the generated gradient waveforms. These may include feedforward correction models that account for coil heating and system dynamics [14], or closed-loop feedback systems that directly measure and adjust the output in real time [15]. In addition, preemphasis techniques are often applied to reduce the impact of eddy currents, particularly in high-speed or high-resolution imaging.

In multichannel spectrometer architectures, gradient waveform generation is extended to support multiple independent output channels. This is particularly important for gradient array systems, where each coil element may require a unique waveform to generate spatial encoding magnetic fields.

To address these demands, we developed a scalable architecture consisting of a central control unit and a set of gradient driver chains [15]. Each chain integrates a digitally controlled waveform generator with embedded feedback compensation and drives four full-bridge amplifiers configured for array operation.

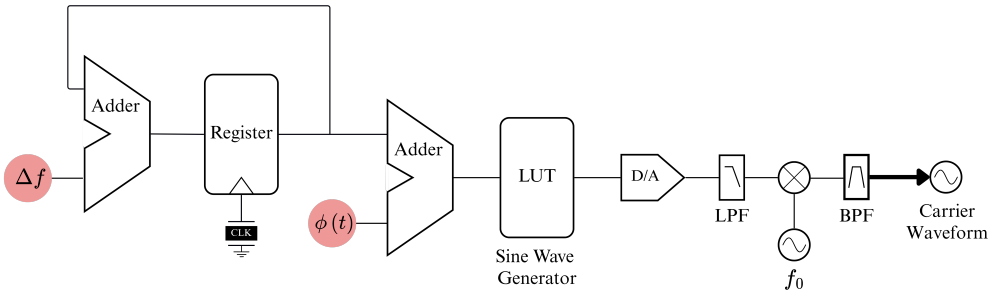


Figure 2.8: An example flow diagram for RF carrier synthesis

2.3.4.3 RF Transmitter

The RF transmitter subsystem generates the excitation signal to manipulate spin magnetization during an MRI pulse sequence. The signal must be carefully controlled in frequency, phase, and amplitude to achieve the desired excitation profile. Modern systems produce this signal digitally, offering high precision and flexibility.

In many digital RF transmitter designs, the waveform is not generated directly at the Larmor frequency. Instead, it is first synthesized at a lower baseband frequency and then shifted up to the target frequency through mixing. This two-stage process depicted in Fig. 2.8 provides better control over the signal characteristics and allows the use of higher-resolution digital-to-analog converters (DAC), which are often limited in performance at high output rates.

At baseband, the waveform is modulated according to the phase and frequency settings defined by the pulse sequence. Since the signal evolves more slowly at these frequencies, the DAC can output more samples per cycle, resulting in higher resolution and improved waveform smoothness. This approach mitigates staircase artifacts and reduces spectral distortion that can occur in direct high-frequency synthesis with limited sampling depth.

The baseband waveform is generated digitally using an NCO, which accumulates phase and offset frequency information over time using a digital adder.

This accumulated phase is then passed to a lookup table (LUT) that maps each value to the corresponding amplitude of a sinusoidal waveform. The resulting digital samples are sent to a DAC and smoothed by a low-pass filter for analog conversion. As a final step, the encoded baseband waveform is mixed with a high-frequency carrier from a local oscillator or a separate direct digital synthesizer (DDS) channel to shift the signal to the Larmor frequency.

The advantage of this architecture lies in its modularity. Phase, frequency, and amplitude components can all be modulated independently in the digital domain before upconversion. Additionally, the digital baseband signal can be easily shaped using envelope modulation or gating logic, and phase cycling strategies can be embedded directly into the accumulator logic. For these reasons, baseband-driven digital RF synthesis remains the dominant approach in many modern MRI spectrometers, especially those implemented on FPGA platforms.

While the majority of RF transmitter designs rely on DAC-based analog synthesis at baseband, the system developed in this work adopts a novel approach to RF generation. Instead of using a DAC to convert sinusoidal waveforms, the carrier signal is synthesized and filtered using only the FPGA resources. This method avoids the resolution limitations of DACs at high frequencies and enables direct control over phase and frequency components through fractional clock division. The envelope of the RF pulse, read from the sequence memory, is concurrently generated by a separate module in the form of digital PWM signals. The resulting architecture enables precise and reconfigurable RF waveform generation using standard FPGA logic without the need for high-speed analog components. Furthermore, by requiring no external components, the proposed methodology provides a low-cost, scalable solution.

2.3.4.3.1 Fractional Clock Divider Fractional clock dividers allow the generation of non-integer output frequencies from a fixed input clock, enabling precise, high-frequency carrier synthesis directly on an FPGA[1]. In our case, this

[1]The idea of using fractional clock dividers as RF synthesizers was inspired by the blog of Dan Gisselquist on [42].

modality is quite valuable since it allows us to produce RF frequencies such as 123.238945 MHz from a 100 MHz system clock with minimal resource expenditure.

The core idea relies on a digital accumulator that increments by a fixed value each cycle. The MSB of this accumulator toggles pseudo-periodically, resulting in an output with an average frequency given by:

$$f_{\text{out}} = f_{\text{clk}} \cdot \frac{\texttt{inc}}{2^b}$$

where `inc` is the periodic increment and b is the bit width of the accumulator. While this generates the correct average frequency, the output has significant jitter due to variable edge spacing.

To address this, it is possible to increase the effective clock frequency in a Xilinx Ultrascale model [43] FPGA by serializing the output using the `OSERDESE3` primitive [44]. In each clock cycle, eight delayed versions are calculated instead of one delayed value of the accumulator, and the MSBs of these stages are collected into an 8-bit parallel word. Thus, serialization increases the effective clock frequency to eight times what it used to be, and the apparent phase noise is reduced.

However, even with the increased output clock frequency, the signal still exhibits timing irregularities and duty cycle distortions due to the discrete nature of the accumulator steps. To compensate for this, the serialized stream can be looped back into the FPGA through a clock-capable input pin and routed into a `Mixed-Mode Clock Manager` (`MMCM`) module [42]. This final stage acts as a dynamic filter, locking onto the average frequency while suppressing high-frequency jitter and edge inconsistencies. The cleaned output becomes a stable RF carrier, which is then used directly by the RF driver for pulse generation.

This combination of accumulator-based division, oversampling, and phase-locked-loop (PLL) filtering enables fractional clock dividers to serve as viable and flexible RF clock sources within FPGA-based systems.

Overall, this architecture offers a fully digital and tunable RF source, implemented entirely on the FPGA without relying on analog mixers or high-speed DACs.

2.4 Overview of Gradient Array Technology

Conventionally, MRI hardware is based on three large gradient coils placed along the x, y, and z axes for spatial encoding. These coils are fixed in geometry and are driven independently to generate linear magnetic field gradients along each direction. This architecture has remained largely unchanged for decades and serves as the foundation of virtually all commercial MRI scanners. It enables reliable imaging in a Cartesian coordinate framework and is well-supported by established hardware, safety models, and reconstruction algorithms. However, this setup introduces several intrinsic limitations:

- **Fixed spatial geometry:** The gradient coils are large and fixed in orientation; therefore, imaging is restricted to the isocenter of the scanner, where the gradient fields are most accurate. To access different anatomical regions, the patient bed must be physically repositioned, limiting flexibility and increasing setup time in multi-region exams.
- **Gradient nonlinearity at peripheral regions:** Although the gradient fields are designed to be linear near the isocenter, their linearity degrades toward the edges of the field of view. This leads to reduced spatial encoding accuracy, particularly in off-center imaging applications such as musculoskeletal or whole-body imaging.
- **Concomitant fields:** When gradient fields vary spatially and temporally, they inevitably generate second-order magnetic field components—commonly referred to as concomitant fields. These nonlinear terms are not part of the intended encoding process and can cause unintended phase accumulation and spatial distortions, particularly in sequences with strong gradients or large field-of-view coverage.

- **Peripheral nerve stimulation:** Rapid changes in gradient fields induce electric fields in the body. When these fields exceed certain thresholds, they can cause PNS, especially in regions such as the head or neck. With high-performance MRI systems, the limited degrees of freedom in conventional coil geometry make it difficult to avoid stimulating specific anatomical regions.

- **Eddy currents:** The switching of gradient coils induces eddy currents in surrounding conductive structures, such as the cryostat or RF shields. These eddy currents generate secondary magnetic fields that persist for several milliseconds and can interfere with image quality. While pre-emphasis and compensation methods exist, they are tuned for specific coil geometries and become less effective in off-isocenter or oblique imaging scenarios [45].

To overcome these challenges, multi-coil systems have emerged as a promising alternative to conventional MRI gradient systems. Early implementations, such as the modular gradient coil concept, aimed to divide the gradient generation task among multiple independently driven sub-coils. Harvey and Katznelson pioneered this approach in a series of theoretical, experimental, and patent contributions [1, 2, 6, 46]. Their designs enabled localized gradient generation with reduced PNS, improved linearity, and enhanced diffusion-weighted performance even at moderate field strengths. Parallel developments explored multiple-region gradient arrays, providing extended field-of-view and spatial flexibility by partitioning the bore into independently controlled zones [3]. Additional advances introduced continuously variable field characteristics [5], localized gradient strategies [4, 47], and non-Cartesian encoding geometries such as PatLoc (Parallel Imaging with Localized Gradients) [7–9]. These multi-coil systems—later termed matrix coils or gradient arrays—offered the ability to dynamically synthesize linear and nonlinear encoding fields, paving the way for anatomically adaptive, region-specific imaging protocols.

Later, gradient array coils with higher channel numbers were demonstrated. Ertan et al. replaced the z-coil of an existing 3T MRI with a custom-made 9-channel gradient array system and simultaneously excited multiple slices with a

single-band RF pulse [12]. Littin, in his dissertation thesis, built an arbitrarily positioned 84-channel coil system [13]. By inserting it into a 3T MRI system, he applied additional highly localized fields to manipulate the phase and frequency encoding gradients and obtain a multi-slice image within a single imaging frame. However, his design had a limited number of gradient power amplifiers (GPAs) and was forced to externally connect individual gradient coils. As a result, arbitrary coil configurations could be realized, but the system was effectively limited to 12 channels.

More recently, the increased degrees of freedom offered by multichannel gradient arrays have been targeted to investigate the trade-offs between gradient field linearity, PNS thresholds, and cryostat-induced power losses due to eddy currents. Takrimi et al. [48] demonstrated that manipulating the shield geometry in a dedicated actively shielded array coil can improve field containment and reduce eddy current interactions. Additionally, they showed that the shield structure can serve as a secondary source for gradient fields, enabling stronger encoding performance. Building on this, a subsequent study [49] introduced a current distribution optimization strategy aimed at minimizing eddy current power dissipation within the cryostat, while preserving the encoding performance of the primary array. In parallel, Babaloo et al. [50] proposed an optimization framework to minimize the electric fields induced within the body, thereby increasing PNS thresholds without compromising gradient field strength or spatial resolution. Moreover, by relaxing conventional constraints—such as strict linearity of the encoding fields—significant improvements in achievable PNS safety margins were demonstrated.

2.4.1 Gradient Array Sequences

Gradient arrays present a paradigm shift in spatial encoding by replacing the traditional three large, fixed gradient coils with multiple smaller coils distributed around the imaging volume. Each element in the array is independently controlled, allowing for flexible and localized magnetic field generation. As each

channel can be positioned in arbitrary positions, the paradigm shift also affects the pulse sequence design perspective.

Contrary to conventional gradient systems, the Cartesian directionality of the x, y, and z axes can be disrupted in gradient array systems, as shown in Littin's work [13]. The spatial arrangement of array elements can be non-orthogonal, with arbitrary channel positions and unknown magnetic field responses. Consequently, the encoding fields generated by the array are not easily predictable or analytically defined.

Electromagnetic simulation tools must be employed to characterize these fields, and the physical coil geometries must be modeled. One can numerically estimate its corresponding gradient field response by applying a unit current to each channel individually. These responses are assumed to be independent due to current-controlled drivers, which suppress electromagnetic cross-talk between elements.

Once the individual field maps are known, they can be combined into a matrix representation, where each row corresponds to a spatial location restrained under a region of interest (ROI) and each column represents the field contribution of a single channel. This matrix serves as a basis for computing optimal linear combinations of the array elements that approximate a desired gradient field. The optimization is carried out under specific constraints, such as gradient linearity, PNS thresholds, and minimization of eddy current losses.

Solving this optimization yields a vector of scaling coefficients that determine how strongly each channel should be driven to synthesize the target field. By repeating this process for every event in the pulse sequence—such as slice selection, readout, or phase encoding—a time-varying set of current amplitudes for each array element can be generated. Together, these transient amplitudes form the complete gradient array sequence, effectively representing a dynamic reweighting of coil contributions to produce the required encoding fields across time.

In general, following these unconventional design patterns reduces the overall efficiency. Individual small loops act destructively on each other, and high power

must be expended to generate the required gradient fields. Therefore, gradient array coil layouts resemble the spatial arrangement of conventional systems, such that certain subsets of channels predominantly contribute along the x, y, or z directions. As a result, the field responses of individual coil elements become more predictable.

In this semi-structured case, the field responses must still be characterized via electromagnetic simulation or empirical measurement. However, due to the preserved directionality, each conventional waveform (e.g., slice selection or readout gradients) can be directly projected onto the corresponding subset of gradient channels using simple linear scaling. This eliminates the need for full-field optimization across all time points and spatial locations, significantly reducing the computational burden. In contrast, the general case requires voxel-wise field calculations and repeated constrained optimizations for each event in the sequence.

As a result, directionally structured array systems offer a more efficient, realizable, and interpretable framework for implementing gradient array sequences. The optimization process in this case yields a matrix of current weighting coefficients, called optimized current weights, that are used to linearly scale the waveforms of conventional sequences across channel groups. In the remainder of this thesis, this directionally structured array configuration is assumed as the baseline architecture.

2.5 Chapter Summary

In this chapter, we outlined the core background necessary for the rest of this thesis. We started with the basics of MRI physics and then introduced key concepts behind pulse sequence design, including a closer look at oblique imaging and its specific challenges. We continued with a discussion of `PulSeq`, the open-source sequence framework used extensively throughout this work. This was followed by an overview of MRI spectrometers, focusing on the RF transmit chain, given its relevance to later chapters. Lastly, we introduced gradient array technology

and described the principles behind gradient array pulse sequence design. These topics form the foundation for the architectural and implementation work that follows.

Chapter 3

Spectrometer Architectures

This chapter presents the spectrometer architectures developed to control multiple gradient coils and parallel RF transmit arrays with dynamic phase modulation. All subsystems are implemented entirely on the KCU 105 Ultrascale [43] FPGA evaluation board and designed with modularity and scalability. Each subsystem operates independently but can be synchronized with others through trigger signals and internal control lines. The overall structure of the system is illustrated in Fig. 3.1.

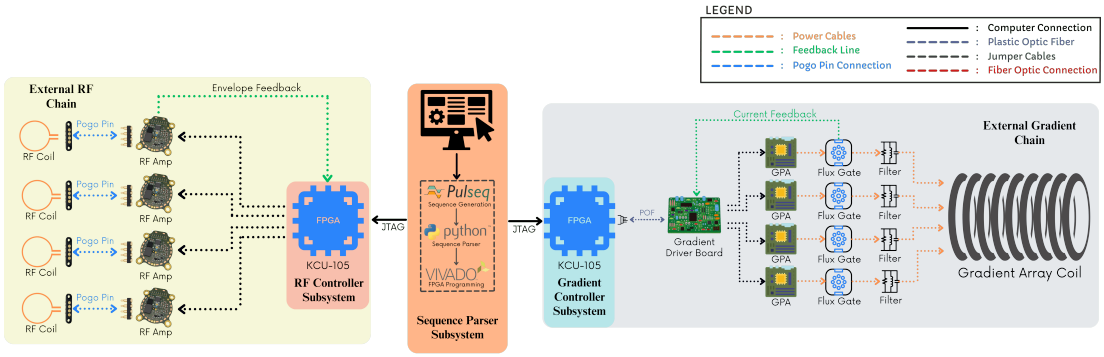


Figure 3.1: Overview of the combined gradient and RF controller subsystems.

The architecture is composed of three primary subsystems:

- A **sequence parser**, which reads PulSeq (v1.4.0)-formatted sequence files

and converts them into a hardware-compatible format for FPGA memories;

- A **gradient controller**, which reads the parsed sequence, generates time-resolved gradient amplitudes, maps them onto array channels, and transmits them to external driver boards to drive an array of independently controlled gradient coils;
- An **RF controller**, which synthesizes both high-frequency carrier signals and low-frequency envelope modulations according to the parsed sequence file and drives an array of RF transmitters with dynamic phase shifting capabilities.

Each subsystem has been designed to operate in isolation or as part of the integrated system, allowing the spectrometer to be used in standalone configurations for subsystem-level testing, prototyping, or targeted experiments. In the following sections, the design and implementation of each subsystem are described in detail.

3.1 Pulse Sequence Parsing and Data Preparation

As described earlier in Chapter 2.4.1, the array configuration used in this thesis emulates conventional gradient axes using multiple smaller coils, each treated as an independent channel. Therefore, traditional pulse sequence design methodologies can still be applied. Once designed, the sequence file can be put into the parser without considering the array structure. The gradient controller handles the projection onto the array structure, as mentioned later in Chapter 3.2.

Pulse sequences are developed using the PulSeq framework due to its vendor-independent and low-level sequence file format. Its hierarchical organization, particularly under the `[BLOCKS]` structure, enables navigation through components via identification (ID) tags, as detailed in Chapter 2.2.3. However, while

the format promotes human readability and traceability, it introduces non-trivial challenges when parsing the sequence into a hardware-compatible memory structure.

One of the primary difficulties lies in the hierarchical referencing mechanism. Sequence blocks often contain nested IDs, forming dependencies that begin at `[BLOCKS]` and traverse downward into substructures such as `[RF]`, `[GRAD]`, and eventually `[SHAPES]`. Unwrapping the hierarchy requires the parsing to start from the lowest level and gradually ascend to ensure correct memory referencing.

Further complications arise from PulSeq-specific object definitions and distinctions. For instance, the difference between a generic gradient event `[GRAD]` and a trapezoidal event `[TRAP]`, or the compressed versus uncompressed waveform data in the `[SHAPES]` section, demands different treatment during decoding. Each object type has its own metadata structure and waveform representation, which altogether increases the complexity of the task.

Here, one may consider simplifying the architecture by allocating fixed memory blocks to specific event types. Still, such an approach is impractical due to the highly variable structure and size of different sequences. Instead, a more flexible and dynamic parsing strategy is required to map the sequence hierarchy into the FPGA memory space correctly.

3.1.1 Sequence Parser Subsystem

The sequence parser subsystem consists of a Python-based script that reads a sequence file with the ".seq" extension, assigns memory addresses and pointer relationships between sequence objects, and generates memory configuration files such as TCL scripts and simulation headers.

3.1.1.1 Mapping Sequence Objects to Memory Addresses

Our implementation strategy follows a hybrid approach that combines both top-down and bottom-up traversal of the sequence hierarchy. This is necessitated by the nested referencing structure of the PulSeq format, where higher-level objects (e.g., `[BLOCKS]`) point to lower-level components (e.g., `[RF]`, `[GRAD]`, `[SHAPES]`) by ID, while the actual memory placement of these components must be resolved in reverse order.

The parser begins with the top-level `[BLOCKS]` section, which defines the temporal structure of the pulse sequence. Each block entry contains seven fields: total duration, `RF` ID, gradient IDs along the three logical axes (`GX`, `GY`, `GZ`), `ADC` ID, and `EXT` ID. These fields reference objects defined elsewhere in the sequence file (see Chapter 2.2.3).

During the top-down phase, the parser descends through the hierarchy and inserts placeholders in place of unresolved IDs. These placeholders reserve memory slots for objects that are yet to be stored but ensure that the overall structural consistency of the sequence is maintained.

Once the parser reaches the lowest hierarchical level—typically the data in `[SHAPES]`—definitive memory addresses are assigned. At this stage, an additional parsing step is required to resolve the ambiguity between compressed and uncompressed waveform data. Since PulSeq does not explicitly indicate these formats, the only way to distinguish them is by comparing the declared *number of samples* against the actual number of stored values. Performing such checks on hardware would be inefficient; therefore, the distinction is encoded during parsing. If the waveform is found to be compressed—that is, if the number of stored points is fewer than the expected samples—the *number of samples* field is marked with a negative sign. This sign acts as a flag, allowing the hardware to determine the waveform format without any additional logic during execution.

Following this, the parser initiates a bottom-up unwrapping phase. The placeholders previously written into the higher-level tables (e.g., `[GRAD]`, `[RF]`) are

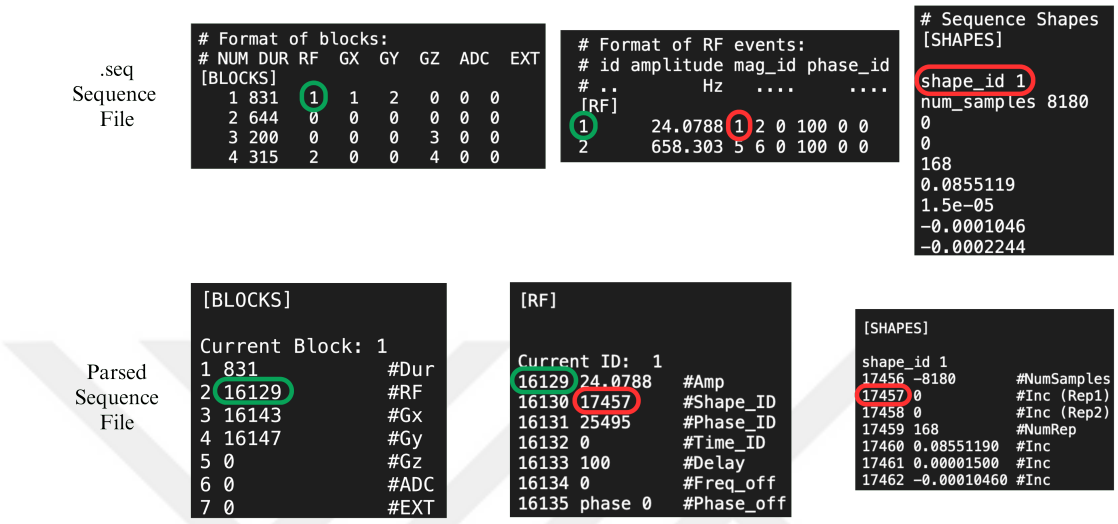


Figure 3.2: An example sequence file segments (top row) and the resulting parsed file format with assigned address values (bottom row)

replaced with the correct memory addresses of their referenced waveform data or event structures. For example, once a shape is stored in memory, its address is inserted into the object that references it. However, as the parser ascends further, another ambiguity of referencing between the `[GRAD]` and `[TRAP]` entries arises. Although both define gradient events, they differ structurally and must be handled separately in hardware. To resolve this, a similar marking strategy was applied: positive addresses indicate a `[GRAD]` reference, while negative addresses are reserved for `[TRAP]` objects.

This two-phase resolution ensures accurate hierarchical resolution and flexible memory mapping with minimal memory expenditure. It accommodates variability in sequence length, object types, and waveform complexity without requiring rigid memory segmentation. An example sequence file and its parsed counterpart can be seen in Fig. 3.2.

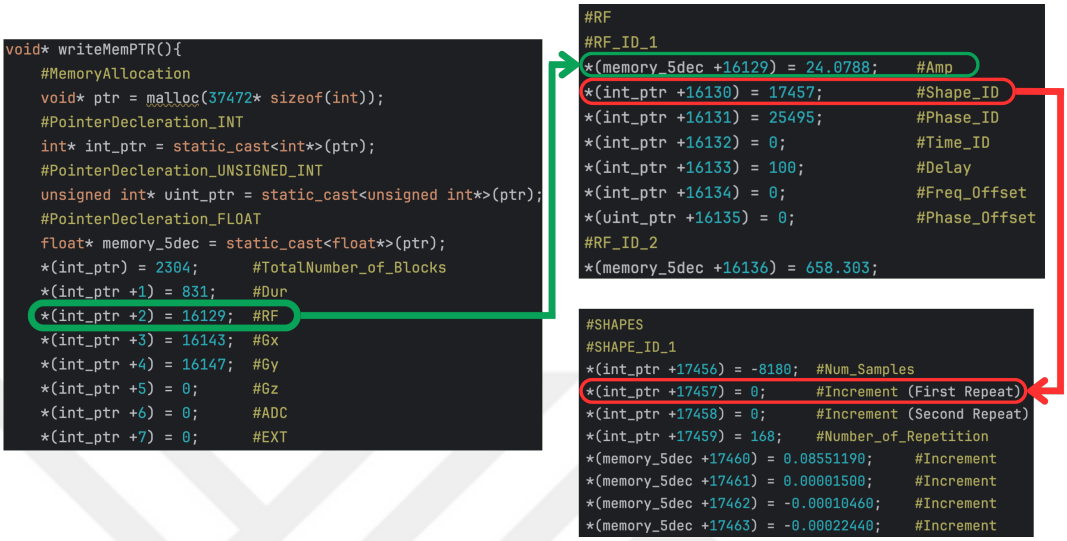


Figure 3.3: Code snippet from an example header file. Different data types can be observed through different pointer definitions. The navigation from [BLOCKS] to [RF] to [SHAPES] can also be traced

3.1.1.2 Memory Mapped Outputs for Simulation and Programming

After parsing and assigning memory addresses to every pulse sequence element, two distinct memory-mapped output formats are generated to support different stages of the development pipeline: simulation environment on Vitis HLS and physical memory programming on the FPGA.

3.1.1.2.1 Vitis HLS-Compatible Header for Simulation The first output is a C++ header file that emulates memory contents in the Vitis HLS environment. This header defines a test bench function that initializes a memory array with hard-coded sequence data. Each entry follows data type conventions from PulSeq and is typed explicitly to reflect structural differences. For example, time vectors in [SHAPES] are stored as integers, while amplitude vectors may include mixed integer and floating-point representations. A code snippet from this header is shown in Fig. 3.3.

```
create_hw_axi_txn write_txn [get_hw_axis hw_axi_1] -type write -address 00000000
-len 128 -force -data {
00000138 000005AC 00000889 00000000 00000000 FFFFF736 00000000 00000000 0000013F
00000890 00000000 00000000 FFFFF731 00000000 00000001 00001249 00000000 00000000
00000000 FFFFF72C 00000000 00000000 00000187 00000897 00000000 00000000 000008BA
00000000 00000000 0000074B 00000000 00000000 00000000 FFFFF72C 00000000 00000000
00000026 00000000 FFFFF727 FFFFF722 00000000 00000000 00000000 00000045 00000000
FFFFF71D 000008BE 00000000 000008ED 00000000 00000045 00000000 FFFFF718 000008C2
00000000 000008ED 00000000 00000045 00000000 FFFFF71D 000008C2 00000000 000008ED
00000000 00000045 00000000 FFFFF718 000008C2 00000000 000008ED 00000000 00000045
00000000 FFFFF71D 000008C2 00000000 000008ED 00000000 00000045 00000000 FFFFF718
000008C2 00000000 000008ED 00000000 00000045 00000000 FFFFF71D 000008C2 00000000
000008ED 00000000 00000045 00000000 FFFFF718 000008C2 00000000 000008ED 00000000
00000045 00000000 FFFFF71D 000008C2 00000000 000008ED 00000000 00000045 00000000
FFFFF718 000008C2 00000000 000008ED 00000000 00000045 00000000 FFFFF71D 000008C2
00000000 000008ED 00000000 00000045 00000000 FFFFF718 000008C2 00000000 000008ED
00000000 00000045
}
run_hw_axi [get_hw_axi_txns write_txn]
```

Figure 3.4: Code snippet from an example TCL script

3.1.1.2.2 TCL Script for FPGA Programming via JTAG The second output is a memory-mapped hexadecimal representation, formatted for direct FPGA programming. Each instruction in the script writes a 32-bit hexadecimal value to a specific address using the JTAG interface. Since each data word occupies 4 bytes, address values are incremented accordingly. For compatibility with JTAG workflows, data are segmented into blocks of 128 entries per write operation. An example TCL output is shown in Fig. 3.4.

JTAG programming was selected for early-stage prototyping due to its simplicity and transparency. Using TCL commands, it is possible not only to write data into on-chip Block RAM (BRAM) or external DDR4 memory via a memory controller, but also to read back specific addresses. This feature was critical in validation to see if the parsed sequence data were correctly written and matched the simulation outputs.

Initially, all sequence data were stored in BRAM. However, due to the limited on-chip memory size, longer or high-resolution sequences could not fit entirely. As a result, DDR4 memory was adopted in later versions, accessed via a Xilinx DDR4 memory controller. While the programming interface remains the same, future system revisions may incorporate PCIe for faster memory loading and support for dynamic reprogramming. Nonetheless, JTAG-based memory initialization has

proven sufficient for most experimental use cases so far.

3.2 Gradient Controller Subsystem

The gradient controller subsystem is responsible for processing the parsed pulse sequence and converting it into gradient array signals suitable for hardware execution. Apart from the memory configuration, every operation from reading the sequence memory to transmitting waveform data to the external driver boards is executed within the FPGA in real time.

3.2.1 Overview and Functional Flow

The gradient controller begins its operation by accessing the sequence file programmed into the memory via the parsed TCL script. For each gradient raster time point, the relevant amplitudes along the three logical axes (G_{ro}, G_{pe}, G_{ss}) are computed. Unlike scanner-based systems, which are constrained to fixed raster times (e.g., 10 µs), the gradient controller enables smoother and higher-resolution waveform generation through interpolation.

The amplitudes and their associated slew rates, which are calculated in the *Memory Interface* submodule, are then passed to an *Array Mapper* submodule. Since the system consists of directional gradient array coils (see Chapter 2.4.1), mapping the conventional gradient waveforms to array channels is rather straightforward. The IP block scales the input amplitudes of three major axes with the precalculated optimized current weight matrix to find the current amplitudes required in the n-channel array configuration.

In the final step of the pipeline, the final array current waveforms are transmitted to an external gradient driver board [15] through the *UART* submodule. Due to the limited data rate of the plastic optical fiber (PoF) communication link, a waveform compression algorithm, similar to the PulSeq compression algorithm,

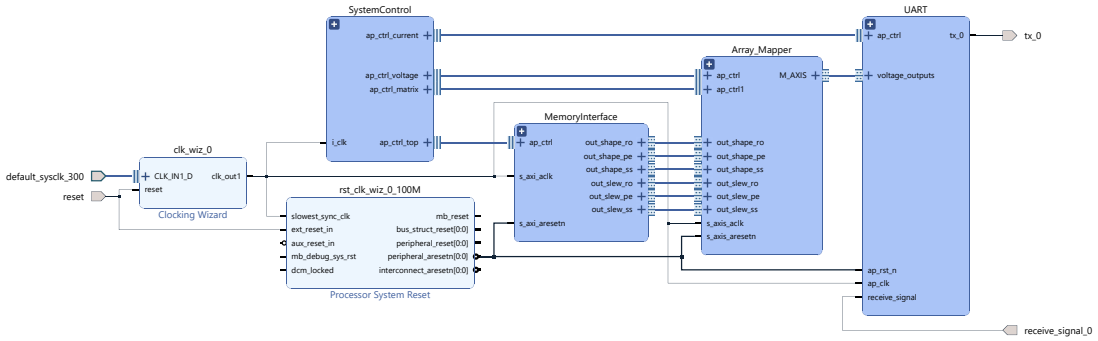


Figure 3.5: FPGA block design of the gradient controller subsystem

is applied before transmission. Thus, the data can always be safely migrated to the gradient driver board within strict time requirements.

The FPGA implementation of the gradient controller subsystem is shown in Fig. 3.5.

3.2.2 Memory Interface Submodule

The *Memory Interface* is a critical component within the gradient controller subsystem. It is responsible for retrieving and preparing the gradient waveform data that is stored in the sequence memory file. All core processing tasks related to memory access are encapsulated in a Vitis HLS-generated IP block named `topfunction`. This IP block manages reading gradient events for the logical axes, parsing both compressed and uncompressed shape data, and evaluating trigger flags when present. The internal structure of the *Memory Interface* submodule can be seen in Fig. 3.6.

3.2.2.1 Topfunction IP Block

The `topfunction` IP block is synthesized from C++ using the Vitis HLS software. It leverages the `#pragma HLS DATAFLOW` directive to concurrently process the information of three logical axes: G_x, G_y, and G_z. This parallel processing

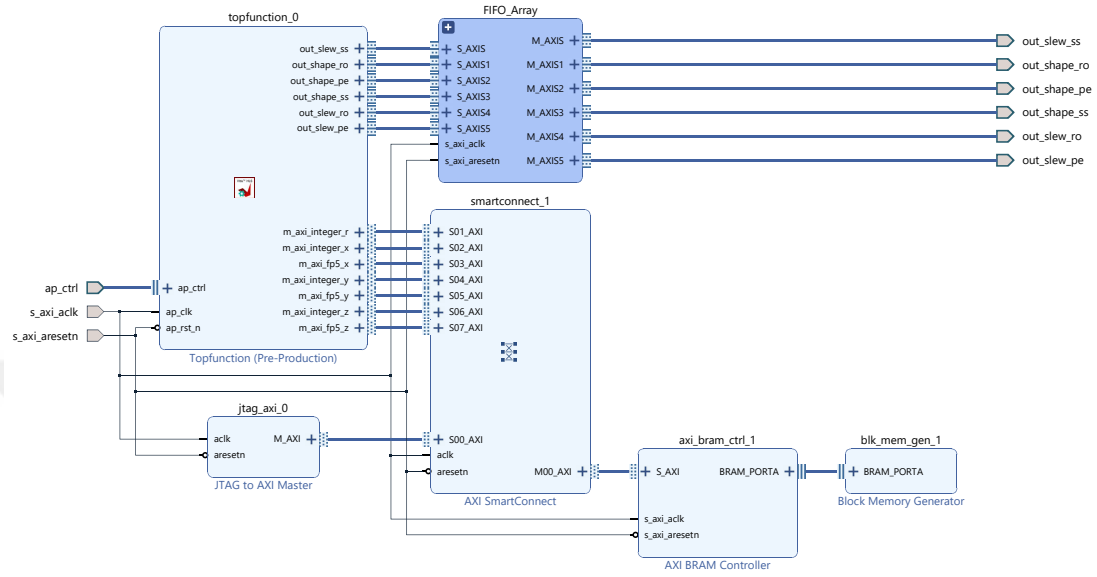


Figure 3.6: The internal structure of the Memory Interface submodule

enables efficient and simultaneous evaluation of gradient amplitudes and their corresponding slew rates at every raster time point.

The format conventions established during sequence parsing allow the `topfunction` IP block to easily distinguish between different gradient events. Each gradient block contains a pointer field indicating the event type it encodes. A **positive pointer** corresponds to a `[GRAD]` event, for which the waveform shape is explicitly stored in memory, and a **negative pointer** signals a `[TRAP]` event, where a trapezoidal waveform is defined by its time values (e.g., rise, flat, and fall times). A **zero** pointer value, on the other hand, denotes a no-operation event: the `topfunction` invokes a subroutine that generates a constant zero output for each raster time until the end of the block duration. This unified pointer convention streamlines conditional logic within the hardware and enables efficient handling of mixed waveform types in a fully pipelined manner. Likewise, the compressed and uncompressed waveform shapes within `[GRAD]` events are distinguished using a dedicated flag that was set during the sequence parsing stage. Based on this flag, the `topfunction` applies the appropriate evaluation strategy—either directly reading the shape data for uncompressed waveforms or performing on-the-fly decompression for compressed shapes.

In addition, the function checks for the presence of external trigger labels defined under the `[EXT]` section of the sequence file. When a trigger is detected, the function appends a special 32-bit flag value of `0xFFFFFFFF` to the data stream. This unique flag, otherwise unreachable in normal data, signals the external gradient driver board to pause execution and wait for an external MRI trigger. Using a simple yet robust signaling scheme, this mechanism allows synchronization between the FPGA-controlled gradient array system and the MRI host scanner.

3.2.2.2 FIFO Buffering and Stream-Based Flow Control

To ensure deterministic timing and isolate individual IP blocks, the outputs of `topfunction` are directed into separate FIFO buffers for each logical axis. These buffers decouple the `topfunction` from the rest of the pipeline, allowing it to store data preemptively.

Each output stream—containing gradient amplitude and slew rate pairs—is written into its corresponding FIFO at a variable rate, depending on the complexity of the gradient shape. For example, the operation is executed rapidly when a channel contains a simple gradient event or no event at all. Conversely, if a complex shape is encountered that requires decompression, the processing time inevitably increases. Without proper buffering, this variation leads to channel mismatch or incomplete waveform generation.

By introducing these stream-connected FIFOs, the design ensures that each downstream block consumes data at a fixed rate defined by the gradient raster time. Continuous real-time operation is guaranteed as long as the upstream blocks (i.e., `topfunction`) produce data faster than it is consumed.

Moreover, using `hls::stream` interfaces provides inherent handshake signaling. If a FIFO becomes full, the write interface stalls the upstream operations until space is available through downstream consumption. This backpressure mechanism automatically halts the `topfunction` without the need for explicit control logic, preventing overflow or timing mismatches.

As a result, the submodule provides deterministic gradient control through a robust memory interface that efficiently evaluates waveforms and facilitates real-time synchronization between external devices and the central MRI hardware.

3.2.3 Array Mapper Submodule

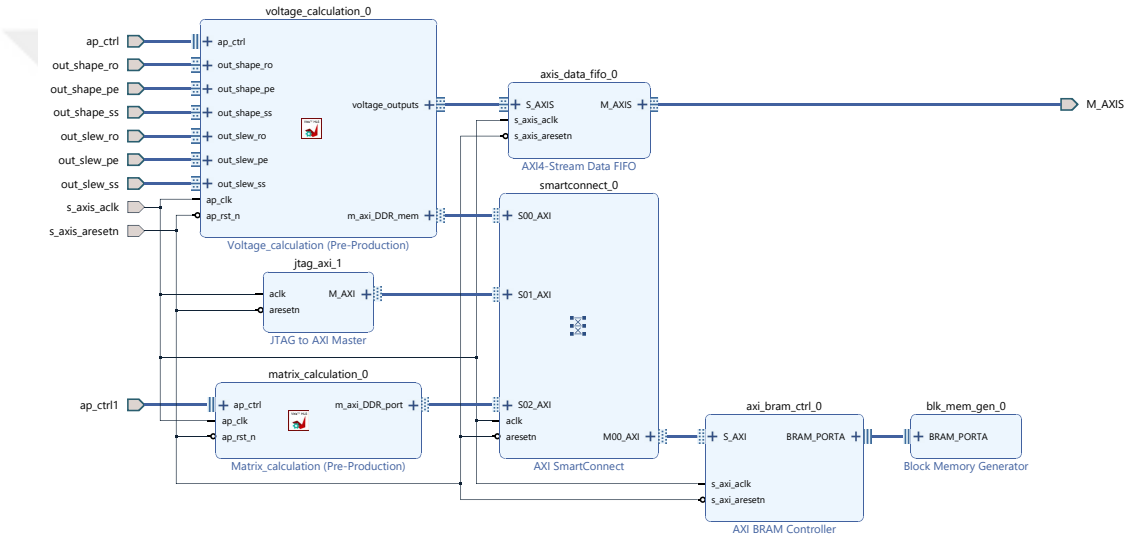


Figure 3.7: Internal structure of the Array Mapper submodule

The *Array Mapper* submodule builds upon prior work by Aydın et al., developed in the context of thermal modeling and feedforward control of gradient arrays in MRI systems [14]. In their design, the gradient array system is modeled using a first-order dynamic model that incorporates not only the core electrical parameters—resistance R, inductance L, and mutual inductance—but also thermal feedback mechanisms to account for heating-induced variations in resistance. The objective is to enable distortion-free current generation through purely feedforward control, thereby eliminating the need for costly feedback circuitry within the gradient amplifiers. This, in turn, can significantly reduce amplifier complexity and the multichannel systems' scaling cost.

In their later work, Aydın et al. [15] developed a gradient driver chain consisting of four full-bridge gradient power amplifiers, each equipped with a digital feedback loop, and a dedicated gradient driver board capable of supporting up to

four such amplifiers. The feedback signals from the amplifiers are routed to the external driver board, where they are processed to adjust the drive signals and ensure accurate current delivery dynamically.

Our current implementation integrates this gradient driver chain, developed in subsequent work of Aydın et al. [15]. With a digital feedback mechanism in place, full feedforward compensation is no longer strictly necessary. Furthermore, the directional structure of the current array coil design simplifies the mapping process even further, as each channel predominantly contributes to a single major axis. As a result, the conventional gradient amplitudes can be linearly projected onto the array channels using a precomputed current weight matrix.

During regular operation, the *Array Mapper* submodule receives the conventional gradient amplitudes from the preceding *Memory Interface* submodule. These amplitudes are linearly scaled using the optimized current weight matrix, which is programmed into the block's internal memory via a dedicated JTAG interface. The resulting n-channel array current amplitudes are concatenated and streamed into a FIFO buffer, ensuring proper timing decoupling and deterministic downstream operation.

Despite these simplifications, the submodule retains full architectural support for the original feedforward modeling approach, including resistance and inductance matrices. This flexibility allows the design to be extended in future studies to support deterministic feedforward control in systems where amplifier feedback is unavailable, undesirable, or costly.

3.2.4 UART Submodule

As the final stage of the pipeline, the *UART* submodule is responsible for transmitting the calculated and mapped array current amplitudes to the external gradient driver board [15]. However, the physical layout of the array system introduces several challenges that must be addressed.

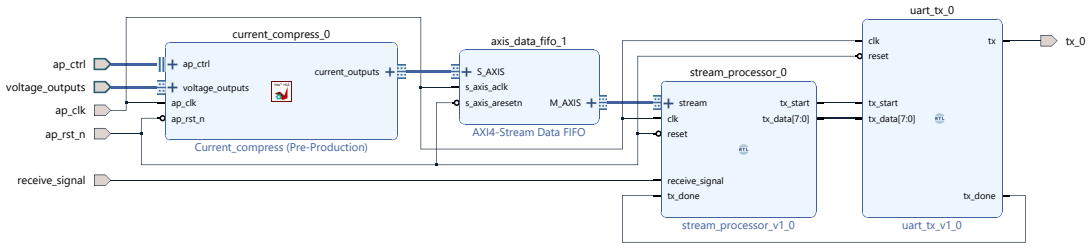


Figure 3.8: Internal structure of the UART submodule

In the current array system, the gradient driver chain is a significant source of electromagnetic interference (EMI), which can damage the nearby sensitive electronics, such as the FPGA. Since the system supports multiple gradient channels and multiple gradient driver chains, the EMI problem scales accordingly. To mitigate this, the gradient driver electronics can be enclosed in a shielded capsule and physically separated from the main spectrometer. However, despite this spatial isolation, a robust communication line is still required over relatively long distances.

Optical transmission provides an effective solution. While high-speed transceivers such as the SFP+ modules on the KCU105 evaluation board [43] could theoretically work, they are costly and impractical for a scalable multi-channel setup. Instead, we adopt low-cost, modular PoFs, which interface with the system through a custom-designed PCB [15]. This configuration enables reliable and EMI-free communication with multiple gradient driver chains, while preserving cost-efficiency and scalability.

However, the PoF transmission line imposes a constraint on the maximum achievable bandwidth. In particular, the data rate is insufficient to support uncompressed real-time transmission of four-channel current waveforms at every gradient raster time. To overcome this limitation, a lightweight data compressor is integrated into the *UART* submodule alongside the standard UART communication IP blocks.

The compression strategy is conceptually similar to the PulSeq waveform compression algorithm (see Chapter 2.2.3.1.4.1) and is designed to reduce the data

payload without sacrificing temporal or amplitude accuracy. By applying this compression scheme before transmission, the system ensures continuous run-time communication without exceeding the throughput limits of the PoF interface.

The resulting compressed data stream is serialized and transmitted via UART to the external gradient driver board, where it is decompressed and used to drive the full-bridge gradient amplifiers. With this architecture, the channel amplitudes can be reliably transmitted to the external driver board within the strict timing requirements of MRI sequence execution.

3.3 RF Controller Subsystem

Earlier work by Arslan [16] inspired the development of this subsystem, who designed a 600 W on-coil Class-E RF power amplifier and its corresponding controller for 3T MRI applications. His design demonstrated precise dynamic phase shifting with an angular resolution down to a single degree. However, a critical limitation of that setup was its dependence on external signal generators to produce the carrier waveform. The carrier needed additional buffering for the Class-E amplifier to work. While it could suffice for benchtop validation, it limited the integration and scalability of a Class-E architecture.

To overcome this constraint and efficiently control the external Class-E amplifier-based RF transmit arrays, which use digital input signals to drive and modulate the gate signals, we developed a dedicated RF controller capable of generating its own digital high-frequency carrier signals. The proposed *RF Controller Subsystem* is a self-sufficient and scalable solution that supports dynamic phase control while maintaining full compatibility with RF transmit arrays.

This chapter presents the architecture and implementation of the RF controller in detail. We begin by introducing the digital RF clock generation strategy based on fractional clock division. We then describe how RF frequency, phase, and envelope data are extracted from the parsed PulSeq sequence and used to synthesize

modulated RF waveforms. Lastly, we briefly mention the integration of the dynamic phase shifting capabilities, which are inherited from the work of Arslan [16]. Together, these components constitute a fully digital RF signal generation chain capable of driving RF transmit arrays in real-time MRI experiments, without relying on external RF sources.

3.3.1 Overview and Functional Flow

The RF controller subsystem begins its operation by accessing the sequence file programmed into memory via the parsed TCL script. Unlike the gradient controller, RF clock generation requires a distinct set of parameters, including transient phase and amplitude values, static phase and frequency offsets, and the center of the RF waveform. While static parameters are singular for each RF pulse, the transient phase and amplitude values are evaluated at each time step. Here, similar to the gradient subsystem, the RF controller is not constrained by conventional raster times (e.g., 1 μs) and can operate at much finer time resolutions, down to the FPGA's internal clock period (e.g., 10 ns), enabling high-resolution transient phase control.

Once the necessary RF parameters are extracted, they are streamed to two dedicated submodules: carrier-related parameters are sent to the *RF Synthesis*, while envelope-related information is passed to the *Envelope PWM* submodule.

Carrier signals are generated by the *RF Synthesis* submodule. A 10 MHz reference clock is received and used as the timing base to ensure synchronization with the external MRI system. This reference clock is internally upscaled and distributed to the RF clock generation components. Because the incoming reference clock does not necessarily align with the FPGA's internal memory or processing clock domains, a clock domain crossing mechanism is implemented using FIFO buffers. Once the carrier-related parameters are buffered, they are adapted to the internal clock domain and forwarded to the `Fractional Clock Divider` IP block. This module reads the phase and frequency offset data associated with the current RF waveform and synthesizes an unfiltered, jittery digital carrier signal.

To improve the resolution of RF frequency and phase characteristics, the raw clock is passed through an `OSERDESE3` serializer, increasing the output rate to 800 MHz by performing 8:1 serialization. This produces a high-speed digital signal that reflects the intended phase behavior but still contains high-frequency jitter and duty cycle distortion. Therefore, the serialized output is routed back to the FPGA's `MMCM` module via a clock-capable pin, where it is filtered and reconditioned. After the filtration, amplifier-specific modulation and driver signals are prepared inside the *RF Chain* submodule and transmitted to the RF power amplifiers.

Although this work demonstrates RF synthesis for a single amplifier, the system architecture allows for easy integration of multiple RF channels. As developed by Arslan [16], dynamic phase-shifting modules can be added to support up to four[1] RF transmitters.

Concurrently, the *Envelope PWM* submodule retrieves the transient envelope waveform from the *Memory Interface* and generates a PWM signal for RF amplitude modulation. The PWM frequency is programmable and can be matched to the specific operating characteristics of the RF power amplifier to maximize efficiency.

The FPGA implementation of the RF controller subsystem is shown in Fig. 3.9.

3.3.2 RF Clock Generation

Conventional systems use analog components in conjunction with digital systems to generate RF clocks. As opposed to it, we propose a fully digital architecture to generate the RF clock signals. In its center resides a core formulation that prescribes the carrier function.

$$C_m(t) = sin\left(2\pi f_0 t + 2\pi \Delta f_m (t - t_{0_m}) + \phi_{0_m} + \phi_m(t)\right) \tag{3.1}$$

[1]The limit here is the available number of `MMCM` modules inside the FPGA.

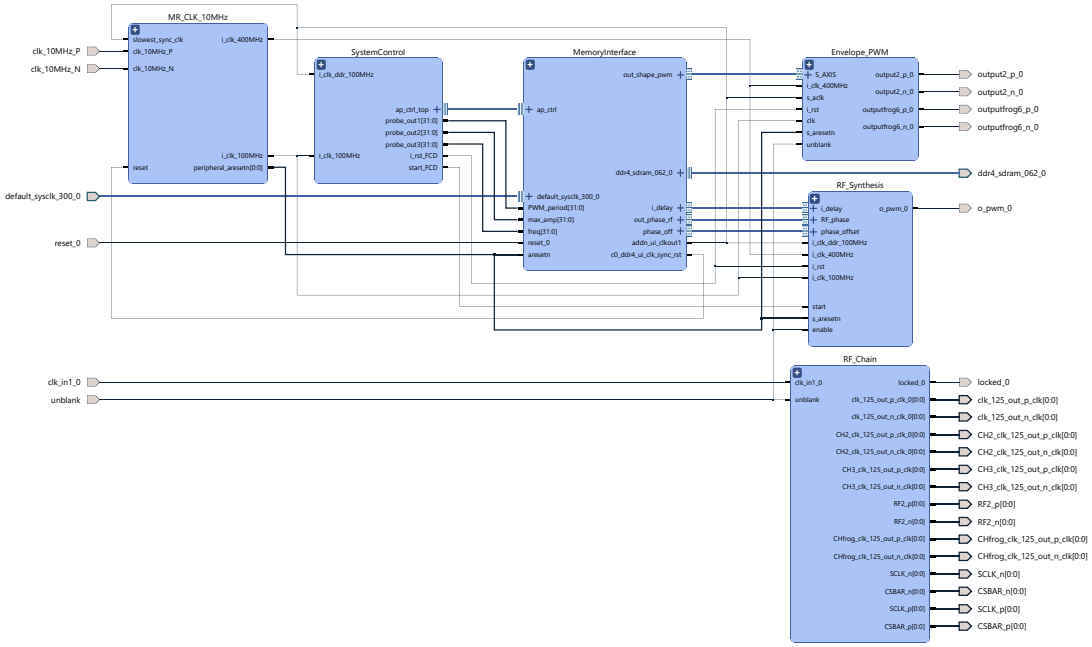


Figure 3.9: FPGA block design of the RF controller subsystem

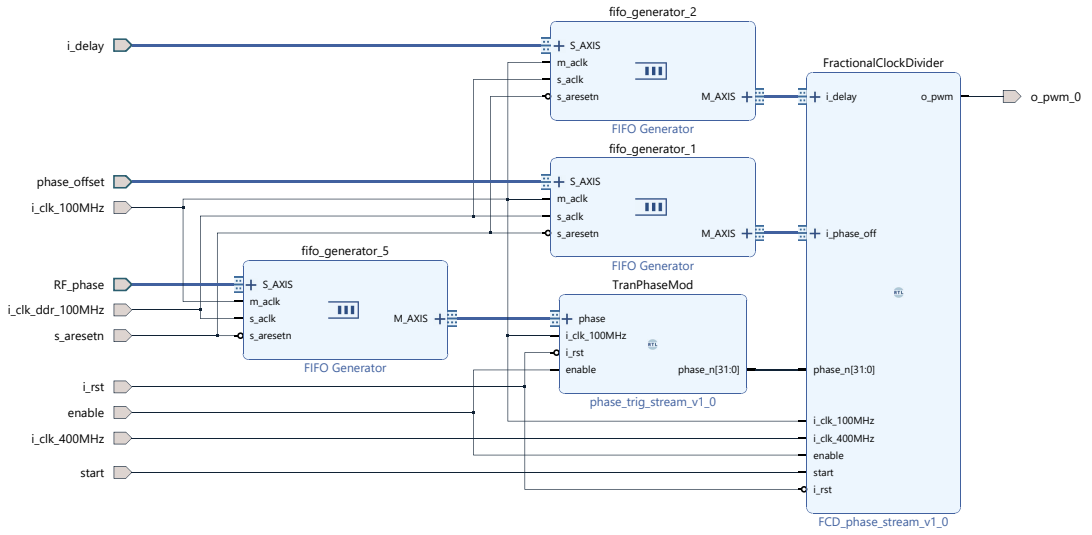


Figure 3.10: Internal structure of the RF Synthesis submodule

3.3.2.1 Formulation of the Digital RF Carrier

Eq. 3.1 shows the general carrier function of $C(t)$ for a single RF waveform placed on an arbitrary position in time, where f_0, t_0, Δf, ϕ_0, and $\phi(t)$ are the Larmor frequency, center of the RF waveform, static frequency and phase offsets, and the transient phase respectively. The subscript m distinguishes each RF waveform throughout the sequence. The carrier function $C(t)$ is only active during the RF events (depicted by $C_m(t)$) and shuts down or is downscaled to an off-band frequency otherwise.

We discretize time as $t = n\Delta t$, where Δt is the output clock period, to transition the formulation to a digital architecture:

$$C_m[n] = sin\left[2\pi f_0 n\Delta t + 2\pi\Delta f_m(n\Delta t - t_{0_m}) + \phi_{0_m} + \phi_m[n]\right] \tag{3.2}$$

Assuming that each RF pulse is geometrically symmetric[2], we define the centered time index as $n\Delta t - t_{0_m} = n'_m\Delta t$, where $n'_m \in \left[\frac{-|RF_m|}{2\Delta t}, \frac{|RF_m|}{2\Delta t}\right]$ and $|RF_m|$ is the length of the m^{th} is the duration of the pulse. The equation simplifies to:

$$C_m[n] = sin\left(2\pi\Delta t(f_0 n + \Delta f_m n'_m) + \phi_{0_m} + \phi_m[n]\right) \tag{3.3}$$

At first glance, the transformations might seem redundant. However, the new digital counters of n and n' have a deeper meaning and ease the conceptualization of the problem.

Here, n denotes the passage of time since the beginning of the sequence. It is continuous and never ceases to increment during the sequence. It tracks the rotating lab frame and allows the carrier function to reset at the beginning of each RF pulse easily. n', on the other hand, is recalculated for each RF waveform. Its

[2]In the rest of the calculations, a geometrically symmetric RF pulse is assumed to obtain a much easier implementation. It is generally hard to find the center of the RF, and even in PulSeq v1.5.0, geometrically symmetric RF pulses are assumed.

main task is to adjust the carrier waveform to have the same phase and frequency as the rotating frame at the RF center despite the frequency offset.

At this stage, multiple implementation strategies can be adopted. In conventional RF systems, a Direct Digital Synthesizer (DDS) is typically used to generate sinusoidal waveforms by computing amplitude values sampled from a digital sine lookup table. These amplitude values are then modulated and upscaled to produce the desired analog carrier signal. However, in the present work, we adopt a fully digital approach by generating the carrier signal directly as a square wave with a programmable duty cycle D, thereby eliminating the need for digital-to-analog conversion.

The continuous phase domain—traditionally defined over the interval $[0, 2\pi)$ radians—is discretized for digital processing to enable this digital square wave generation. Specifically, the full 2π cycle is rescaled to an integer domain $[0, 2^b)$, where the superscript b denotes the number of bits allocated for representing phase values. This rescaling allows the use of fixed-point arithmetic for phase accumulation and comparison, enabling an easy implementation for phase modulation and frequency synthesis within FPGA logic. Thus, the carrier function from Eq. 3.3 can be reformed as:

$$C_m[n] = C_{sq}\left(2^b \Delta t(f_0 n + \Delta f_m n'_m) + \frac{2^b}{2\pi}(\phi_m[n] + \phi_{0_m})\right) \tag{3.4}$$

where C_{sq} is a piecewise function defined for a duty cycle D as:

$$C_{\text{sq}}(t) = \begin{cases} 1, & \text{if } 0 \leq t < D \cdot 2^b \\ 0, & \text{if } D \cdot 2^b \leq t < 2^b \end{cases} \tag{3.5}$$

The number of allocated bits, b, also defines the resolution of the smallest representable frequency for C_{sq}. If Δf_{step} denotes the minimum frequency steps, $2^b \Delta f_{step} \Delta t$ must have an integer value due to discretization. Following this logic, we can find a relation between the number of allocated bits b and the frequency resolution Δf_{step}:

$$b = \left\lceil \frac{\ln(\Delta t \cdot \Delta f_{\text{step}})}{\ln(2)} \right\rceil \tag{3.6}$$

Since the output frequency is 800 MHz, $\Delta t = 1.25 \cdot 10^{-9}$, at least 30 bits must be allocated to accommodate a sub-hertz resolution. In our system, 32-bit values are selected as b values, and the corresponding theoretical resolution for the minimum representable frequency value is calculated as $\Delta f_{step} = 0.186Hz$. With $b = 32$, the function to synthesize digital RF waveforms is ready.

3.3.2.2 Hardware Implementation

For digital implementation, the requirement is to carry out glitch-free calculations on the fly at high frequencies. Under typical operating frequencies, it is possible to design a pipeline to carry out these calculations. However, in our design, the output frequency of 800 MHz is only achieved through serializing 8-bit data. To keep up, the pipeline can be scaled to match the serialization ratio for parallel computation. However, it is challenging to properly route the pipelines to the `OSERDESE3` primitive under strict time constraints. Furthermore, several simplifications can be exploited to make the implementation easier.

First, we define an arbitrary function $g[n]$ in Eq. 3.7, to define the raw result of the carrier function without the sinusoidal or the square wave generator function operators from Eq. 3.1 and Eq. 3.5, respectively.

$$g_m[n] = 2^b \Delta t(f_0 n + \Delta f_m n'_m) + \frac{2^b}{2\pi}(\phi_m[n] + \phi_{0_m}) \tag{3.7}$$

We then use the following points to simplify the implementation:

1. The Larmor frequency f_0, number of allocated bits b, and the time step Δt remain constant throughout the experiment. Therefore, some values can be calculated once and reused for the rest of the operation.

2. Instead of calculating $\frac{2^b}{2\pi}\phi_m[n]$ inside the FPGA, the values can be pre-calculated by the parser to form $\phi'_m[n] = \frac{2^b}{2\pi}\phi_m[n]$. These results can be embedded into memory to remove the need to scale the phase values.

3. The serialization of output values is done inside the `OSERDESE3` primitive. Therefore, the outputs must be calculated in batches of eight[3], in parallel.

4. The transient phase of $\phi(t)$ operates at 100 MHz, contrary to the serialized output frequency of 800 MHz. In other words, the value of the transient phase $\phi[n]$ remains constant for eight time steps. Thus, the increment for each time step inside the serialization can be found as:

$$g[n+1] - g[n] = 2^b \Delta t(f_0 + \Delta f_m) \tag{3.8}$$

5. The constant increment found in Eq. 3.8 remains unchanged during the synthesis of the m^{th} RF pulse. Therefore, it is possible to calculate and store the incremental steps for each serialized output at the beginning of each RF. If the incremental steps are calculated for $k \in [1, 8]$, and stored inside an array, where

$$inc[k] = k \cdot 2^b \Delta t(f_0 + \Delta f_m) \tag{3.9}$$

   then the multiple multiplication/summation operations reduce to the recursive form in Eq. 3.10. For $n \mod 8 = 0$:

$$\begin{aligned} g[n+k] &= g[n] + \text{inc}[k] \\ g[n+8] &= g[n] + \text{inc}[8] + \phi'_m[n+8] \end{aligned} \tag{3.10}$$

   In other words, the target function $g[i]$ can be defined as an accumulator for a more straightforward implementation.

6. Lastly, to redefine each time step as a summation of precalculated values, we can calculate the initial starting point. The contribution of $n' = \frac{-|RF_m|}{2\Delta t}$ at the beginning of the RF pulse can be recalculated as an additional static phase offset as,

$$\phi_{\Delta f_m} = -2^{b-1} \Delta f_m |RF_m|$$

[3]The batch size of eight is due to the 8:1 serialization of the `OSERDESE3` primitive.

In the parser, this value can be embedded inside the original static phase offset of ϕ_0 during the parser calculations. Thus, for the RF pulse that starts at $n = n_m$

$$g_m[n_m] = 2^b \Delta t f_0 n_m + \phi'_m[n_m] + \phi_{0_m} + \phi_{\Delta f_m} \tag{3.11}$$

After these steps are covered, the fully digital RF synthesis can now be realized inside the *RF Synthesis* submodule.

3.3.2.2.1 RF Synthesis Submodule As explained in Chapter 2.3.4.3.1, fractional clock dividers are powerful tools for non-integer frequency division. At their core lie an accumulator and a comparator, both of which we formulated in our previous discussions: an accumulator formulated in Eq. 3.10 and Eq. 3.11, and a comparator in the form of a square wave generator as defined in Eq. 3.5.

The implementation onward is relatively straightforward, but the output is limited by phase noise, which distorts the shape of the generated signal as illustrated in Fig. 3.11. However, the frequency response confirms that the target frequency is still achieved on average, which is the main reason why this method was adopted.

In the *RF Synthesis* submodule, input parameters are streamed from the *Memory Interface* submodule. These inputs are clocked using the 300 MHz DDR4 memory clock, while the RF clock synthesis itself is performed using a separate 10 MHz reference clock derived from the MRI system. To safely transfer data across these two clock domains, FIFO buffers handle the synchronization.

The core component of this submodule is the `Fractional Clock Divider` IP block, where RF waveform generation takes place. Most of the required inputs are preprocessed during the parsing stage to avoid redundant operations and ensure deterministic behavior. Static parameters—such as frequency and phase offsets—are directly provided via the FIFO interface. At the same time, an auxiliary IP block refines transient phase values, `TranPhaseMod`, which increases effective phase resolution through interpolation.

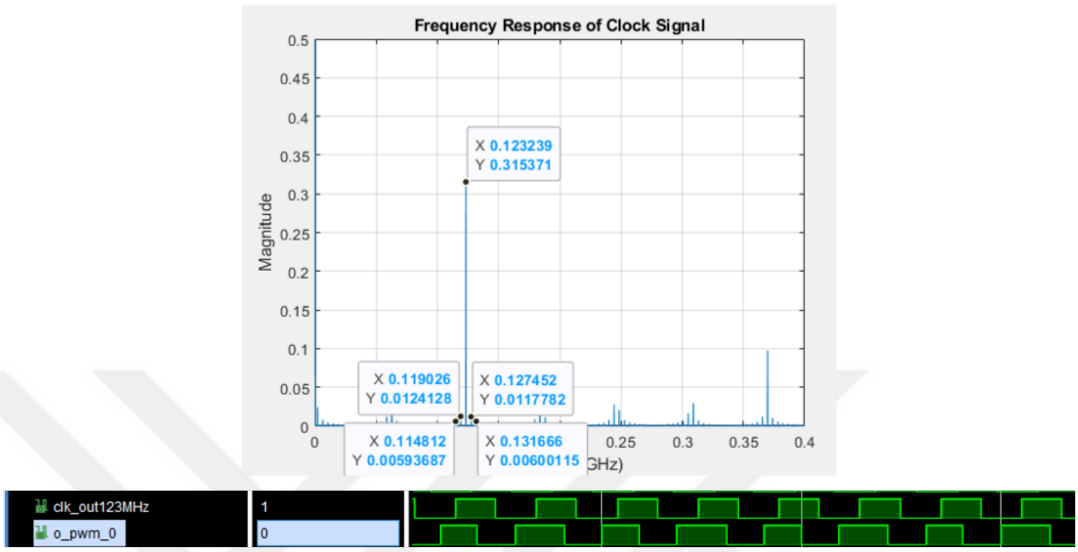


Figure 3.11: The output of the fractional clock divider generating a 123.238945 MHz clock signal (**o_pwm_0**), a reference clock signal (**clk_out_123MHz**) shown at the bottom, and the corresponding frequency response shown at the top.

Inside the `Fractional Clock Divider` block, the synthesis logic follows the formulations given in Eq. 3.10 and Eq. 3.11. The entire synthesis process is event-driven and coordinated by an external *unblank* signal, defining the active RF output intervals. When the *unblank* signal is asserted, the RF clock is synthesized and output using the accumulator logic. When the *unblank* signal is deasserted, the block loads the corresponding static parameters, resets the internal phase accumulator, and reinitializes its state using a reference clock that operates at the Larmor frequency[4].

To generate high-frequency outputs, the synthesized digital clock signal is serialized using the `OSERDESE3` primitive, achieving an effective output frequency of 800 MHz. Several registers are inserted prior to serialization to suppress glitches and maintain timing integrity.

The serialized output is then physically routed back into the FPGA through a

[4]Inside the IP block, a dedicated accumulator acts as a reference counter at the Larmor frequency.

dedicated input pin to serve as the input to the `MMCM` module. This loopback allows the `MMCM` to filter out jitter and duty cycle distortion caused by the fractional division, producing a clean, low-jitter RF carrier signal.

To minimize clock skew and ensure signal integrity, the output register of the `OSERDESE3` is placed on a clock-capable differential pin pair. This reduces trace length and simplifies timing closure in the high-speed routing path.

3.3.2.2.2 RF Chain Submodule The *RF Chain* submodule is where the distorted and jittery clock signal is filtered through `MMCM` modules. The filtered outputs are then passed through several internal components to provide the necessary control signals for the RF amplifiers. Furthermore, the modules for dynamic phase shifting—adapted from the work of Arslan [16]—are also integrated here.

Although the architecture allows phase shifts to be applied during the filtration process, this work demonstrates only a single amplifier output. Therefore, dynamic phase shifting across multiple RF transmit channels is not utilized or displayed. As the RF power amplifiers lie outside the scope of the spectrometer, no further details are provided. The internal structure is shown in Fig. A.1.

3.3.3 Other Submodules

The remaining submodules in the RF Controller subsystem are briefly described below. The detailed descriptions are migrated to Appendix A.1 to avoid cluttering in this chapter.

- **MR CLK 10 MHz**: Receives a 10 MHz clock signal from the external MRI system, filters it, and upscales it to two high-frequency clock signals: 100 MHz and 400 MHz. These clock signals are then used in the *RF Synthesis* submodule.
- **System Control**: Initiates system operations and allows virtual input control for testing parameters such as the PWM output frequency.

- **Memory Interface**: Similar to the submodule described in Chapter 3.6, *Memory Interface* reads the sequence file from memory, extracts relevant information, and streams it to the *RF Synthesis* submodule. The outputs are then passed through FIFO buffers to handle clock domain crossing between the 10 MHz MRI reference clock and the 300 MHz DDR4 memory clock.

- **Envelope PWM**: Reads the envelope amplitude values from the *Memory Interface* and calculates the required PWM width according to the user-defined PWM frequency and system limits. The generated PWM waveforms are then transmitted to the RF power amplifiers to modulate the envelope.

The internal structures of the relevant submodules are shown in Fig. A.2 and Fig. A.3.

3.4 Chapter Summary

This chapter introduced modular and scalable spectrometer architectures that serve as a framework for a self-sufficient, standalone gradient array MRI system. First, we described a dedicated sequence parser that converts PulSeq-formatted sequence files into hardware-compatible memory structures, resolving hierarchical dependencies and encoding waveform formats with minimal memory overhead. Next, we presented the gradient controller subsystem, which reads the parsed sequence, generates time-resolved gradient amplitudes, maps them onto array channels, and transmits them to external driver boards for driving the gradient coils. Finally, we detailed the design of a fully digital RF controller that synthesizes both high-frequency carrier signals and low-frequency envelope modulations according to the parsed sequence file and drives RF power amplifiers. Together, these subsystems contribute to realizing a complete and integrated gradient array spectrometer architecture capable of orchestrated sequence parsing, gradient generation, and RF transmission without reliance on external sources.

Chapter 4

Results of the Controller Subsystems

This chapter presents the results of the implemented spectrometer subsystems and their integration into experimental MRI settings. We evaluate each subsystem independently through simulation and hardware testing,

4.1 Gradient Controller Subsystem

First, the `topfunction` IP block, which is the core function of the *Memory Interface* submodule, is tested. The testbench memory is configured using the header file (see Chapter 3.1.1.2.1), and the IP block is simulated using the cosimulation tool in Vitis HLS. The resulting waveforms displayed in Fig. 4.1 were consistent with the expected sequence profiles.

After the software validation, the output waveforms are tested inside the FPGA hardware. A test function that generated left-aligned PWM waveforms was developed and employed according to the input amplitude. The G_{ro} and G_{pe} channels

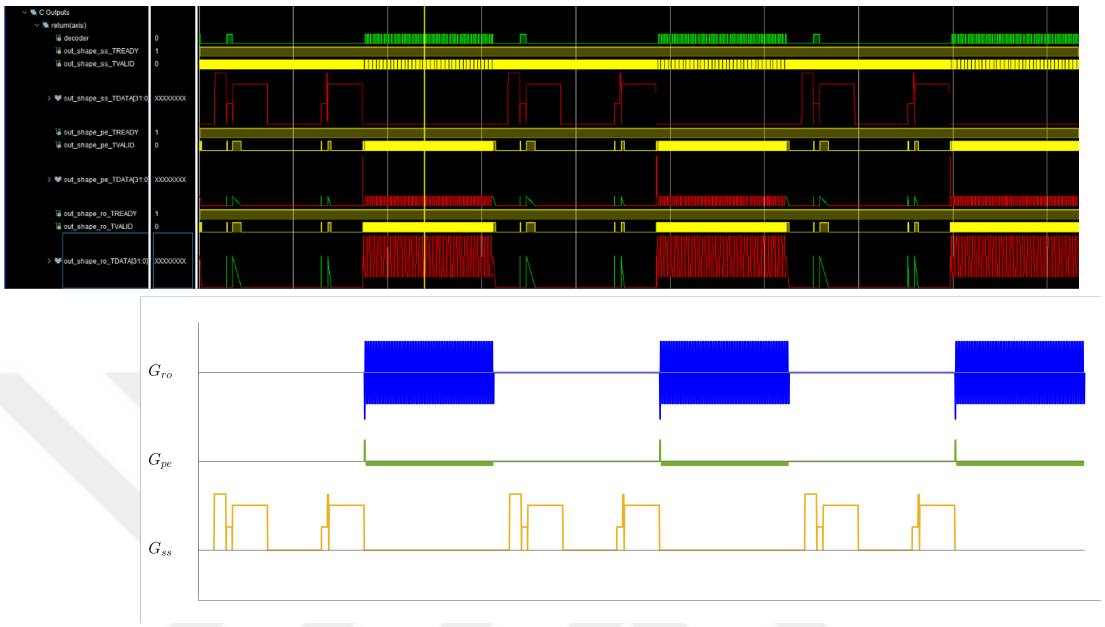


Figure 4.1: Simulated waveforms from the Vitis HLS cosimulation platform (red lines) and the reference sequence plot

from the sequence file shown in Fig. 4.1 were connected to the oscilloscope simultaneously, and the generated amplitudes are compared in a time-resolved manner. The resulting waveforms once again matched the reference as shown in Fig. 4.2.

Since the *Array Mapper* and *UART* submodules are developed as external modules, it was hard to test them over the cosimulation. Instead, the current amplitudes from the testbench simulation were extracted and plotted inside MATLAB. Four additional channels are added to the Z-direction to act as array channels. The corresponding time-resolved amplitudes are obtained by projecting the G_{ss}. The resulting waveforms are displayed in Fig. 4.3. Furthermore, to prove that the system still retains the feedforward model, inductance values are added to the system to generate the corresponding voltage waveforms as shown in Fig. A.4.

After confirming the success of the software architecture, the gradient controller board was physically connected to the external gradient driver board with PoF to test the full gradient pipeline. The external driver board was connected to four full-bridge gradient power amplifiers, and a total of eight flux gate current

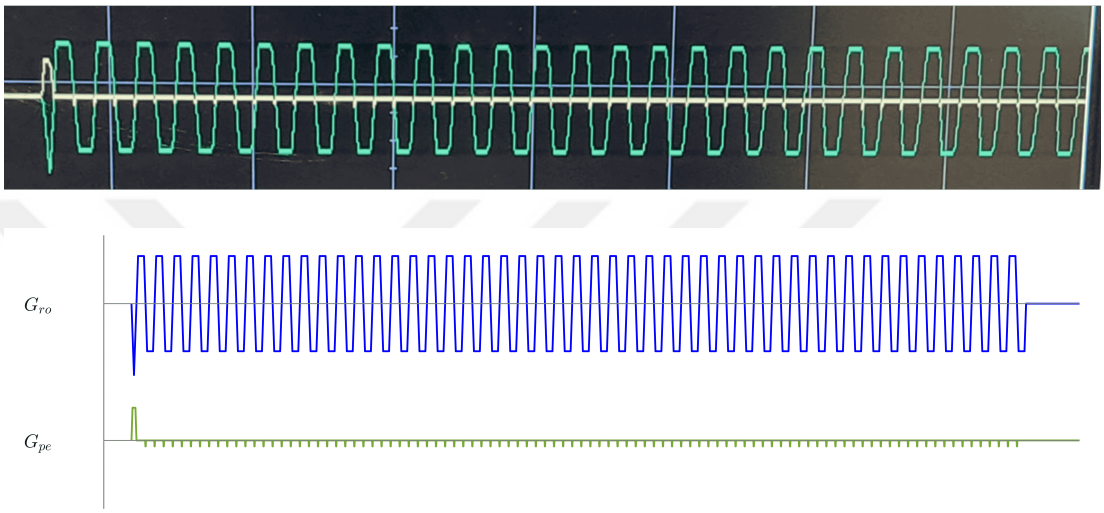


Figure 4.2: Time-averaged PWM waveforms for G_{ro} (green) and G_{pe} (yellow) channels and the reference sequence plot. Due to the low amplitude, G_{pe} appears as tiny ticks on the slopes of the G_{ro}.

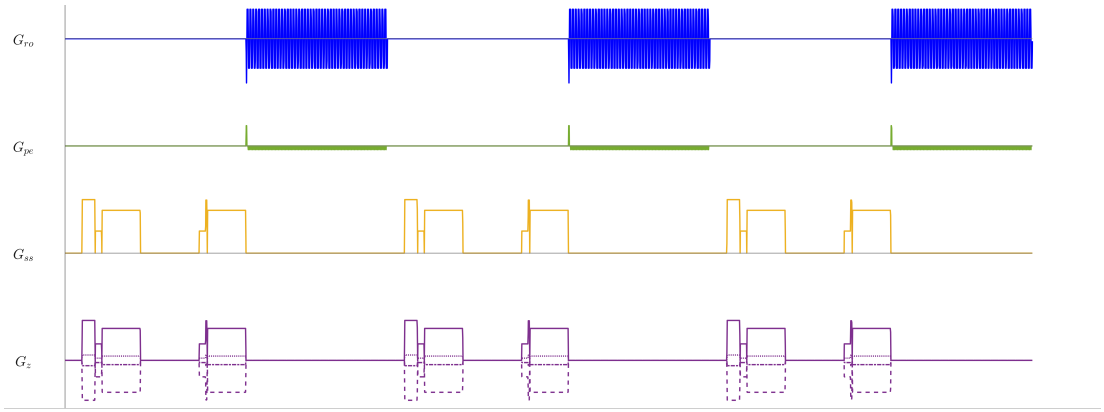


Figure 4.3: Simulation results of the *Array Mapper* submodule. The array currents are projected from the G_{ss}

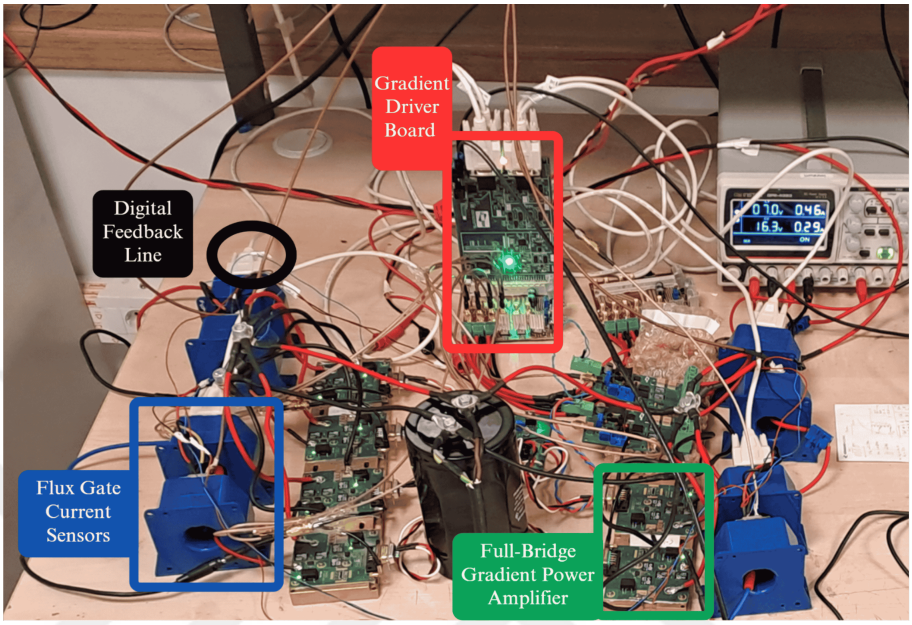


Figure 4.4: Experimental setup of the gradient driver chain. The flux gate sensors in the front are used as digital observers, while the ones on the back provide the feedback line over the RS32 cables back to the gradient driver board.

sensors were placed at the outputs of the full-bridge amplifiers; four were used as the digital feedback channels, and four were connected to the oscilloscope to observe the current waveforms. Finally, the full-bridge amplifiers were connected to four individual coil channels to act as a load. The experimental setup is displayed in Fig. 4.4.

The memory of the gradient controller was programmed with simple trapezoidal waveforms and projected along four channels according to the coil weight matrix P:

$$P = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 0 & 0.8 \\ 0 & 0 & 0.6 \\ 0 & 0 & 0.4 \end{bmatrix} \tag{4.1}$$

Then, through PoF, the projected waveform amplitudes were transmitted to the gradient driver board, which fed PWM waveforms to the amplifiers. The resulting current waveforms are shown in Fig. 4.5.a.

To put the system under more stress and use some of the potential of the

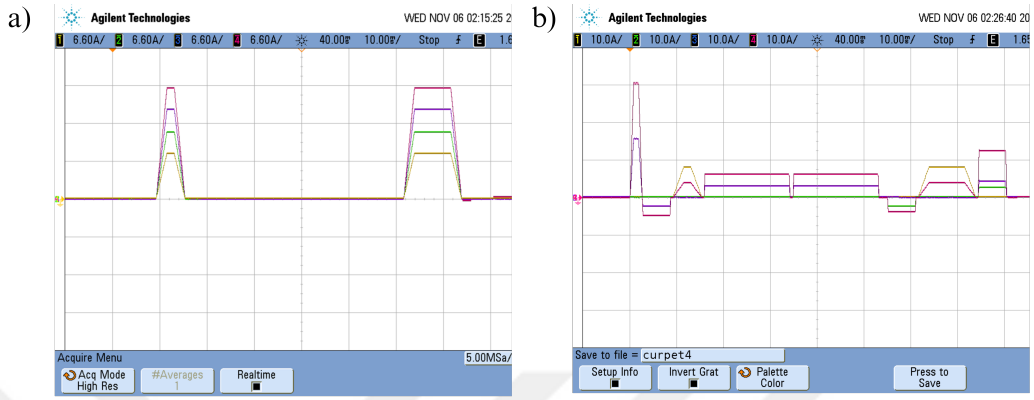


Figure 4.5: Current waveforms generated by the gradient pipeline. a) Simple trapezoidal waveforms with current weight matrix defined in Eq. 4.1, b) A more complex sequence with current weight matrix defined in Eq. 4.2.

gradient array sequences, a more complex sequence shown in Fig. 4.5.b was tested. In the new sequence, the gradients of G_{ro}, G_{pe}, and a combination of G_{ro} and G_{ss} are also included with the current weight matrix P:

$$P = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0.65 & 0 \\ 0 & 0 & 0.5 \\ 0.5 & 1 & 1 \end{bmatrix} \tag{4.2}$$

Once the hardware of the gradient pipeline was tested and validated, the setup was switched to conduct an MRI experiment. A 40-channel custom-built gradient array coil shown in Fig. 4.6 was connected to the gradient driver chain. The current weight matrix was calculated using a double-Maxwell coil configuration to generate a linear gradient field at the isocenter of the coil[22]. As a result, coil weights of

$$P = \begin{bmatrix} 0 & 0 & 0.746 \\ 0 & 0 & 0.1 \\ 0 & 0 & -0.1 \\ 0 & 0 & -0.746 \end{bmatrix} \tag{4.3}$$

were obtained for coil numbers of [1, 4, 9, 12] in the z-direction, respectively.

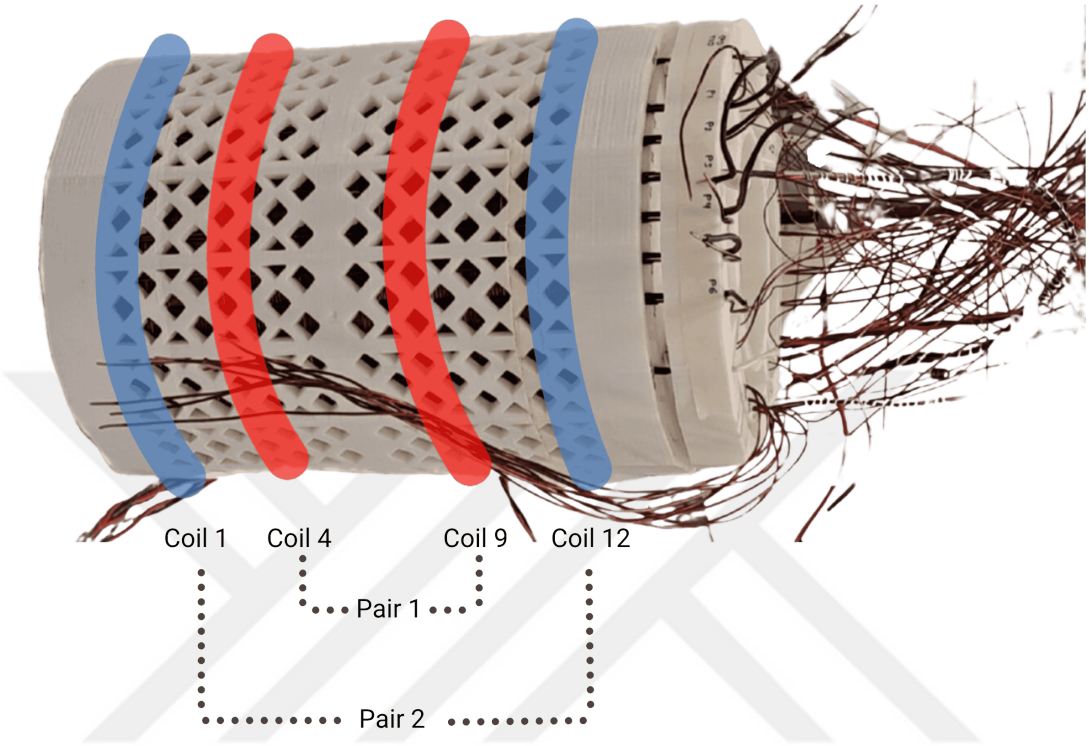


Figure 4.6: Custom-built 40-channel gradient array coil used in the experimental setup. There are 12 primary and 12 shield coils with a total of 24 channels in the z-direction, and four primary and four shield coils with a total of 8 channels in each of the x and y-directions.

Channels 1 and 12 formed the first Maxwell pair, while channels 4 and 9 formed the second pair[1].

The setup for the MRI experiment, as shown in Fig. 4.7, was a hybrid configuration, where the z-direction of the MR coil was disabled and replaced by the custom-built gradient array coil. The gradient controller was placed near the MR console and connected to the gradient chain via PoF. The custom-built gradient array coil was placed inside the MR, on the patient bed, and connected to the full-bridge amplifiers through the back plane.

Two sequence files were provided separately to both the MR console and the

[1]Under normal circumstances, it was possible to short-circuit different potential Maxwell pairs to each other and simultaneously drive eight gradient channels using only four full-bridge amplifiers. However, to validate the stability of the overall gradient pipeline, each coil of a Maxwell pair was driven individually.

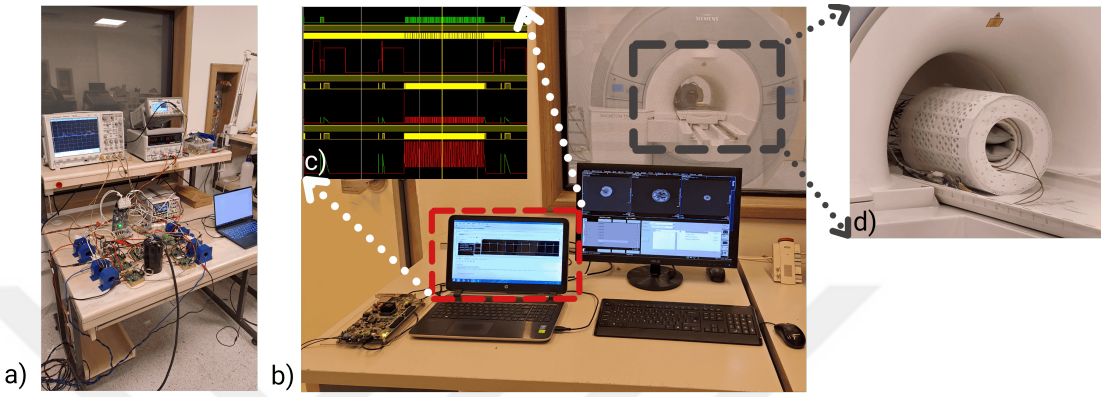


Figure 4.7: Experimental setup, a) 4-channel full-bridge gradient amplifier, b) Programmed FPGA, c) Example sequence plotted on the FPGA, d) A custom-built gradient array coil inside the MRI to serve as the readout gradient.

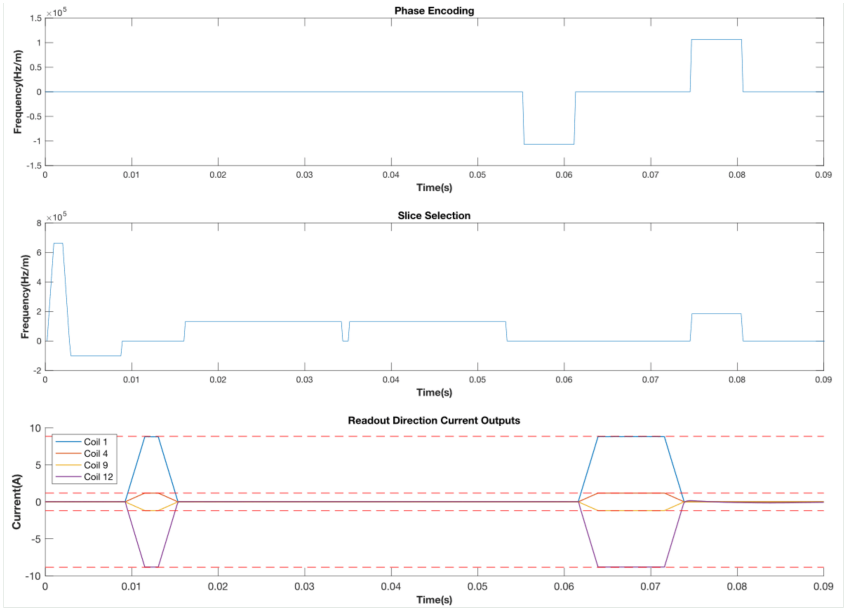


Figure 4.8: Current outputs of separate gradient channels during the experiment. Maxwell pairs show almost identical inverted waveforms. The readout gradients' timing synchronization matches the rest of the sequence.

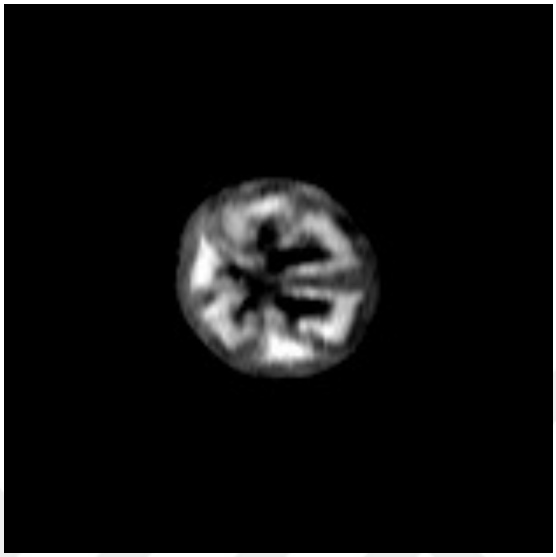


Figure 4.9: The resulting MR image of a tomato phantom obtained from the gradient pipeline

gradient controller. Triggers were employed to synchronize both systems. The imaging plane was flipped to the sagittal plane to further observe the system's stability, where the readout gradient faces the z-direction. A spin-echo sequence with $FOV = 0.5\ mm^2$, $N_x = N_y = 128$, $TE = 12\ ms$, $TR = 26\ ms$, $FlipAngle_1 = 90°$, $FlipAngle_2 = 180°$ was applied, and with the defined hardware configuration, an MR image of a tomato phantom was taken.

The resulting current waveforms of separate channels shown in Fig. 4.8 display the timing accuracy of the time-resolved gradient waveforms alongside the success of the synchronization schema through trigger signals. The acquired MR image also proves the consistency of the waveforms. If there were any imbalance, the image would experience frequency shifts in the frequency encoding domain, but a circular tomato can be seen in Fig. 4.9.

4.2 RF Controller Subsystem

The validity of using a fractional clock divider as the RF clock source was previously tested and confirmed in Chapter 3.3, with the frequency response shown in

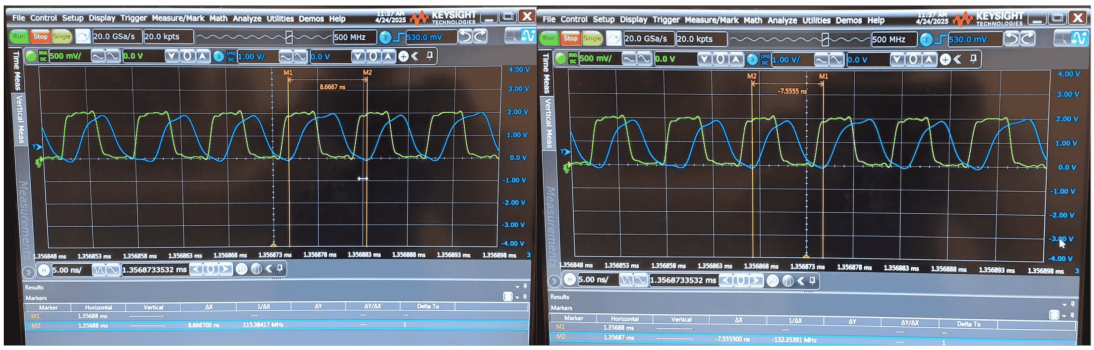


Figure 4.10: Comparison of the raw fractional clock divider output (blue) and MMCM-filtered output (green).

Fig. 3.11. The output of the fractional clock divider was then tested on FPGA hardware, alongside the filtering performance of the MMCM. Two outputs are provided from the clock division, one routed back into the FPGA for filtering, and one to compare as the reference. Both the filtered and raw clock signals were evaluated in terms of phase noise and duty cycle. As Fig. 4.10 displays, the raw output has varying periods, but the filtering from the MMCM maintains a fixed duty cycle throughout the operation.

Because the carrier waveform operated at the Larmor frequency, continuous output signals leaked into the MR room and caused a zipper artifact along a single frequency line. To prevent this, the FPGA was configured to halt operation outside RF events, which in turn forced the MMCM to re-lock to the incoming signal, introducing a delay. The same dual-output setup was used to observe this locking behavior, and the lock time of the MMCM was measured on an oscilloscope.

As shown in Fig. 4.11, the MMCM spends a long time before locking, and it varies for different attempts. However, the MMCM module provides a digital signal named **Locked**, which indicates if the MMCM is locked onto the phase of the incoming signal. Although this aspect was not explored, it is possible to exploit and synchronize the envelope amplitudes according to the **Locked** state.

Afterward, the filtered clock signal was isolated and continuously monitored on an oscilloscope. The performance of the carrier waveform was evaluated and is shown in Fig. 4.12. Although some jitter remained, with a maximum deviation

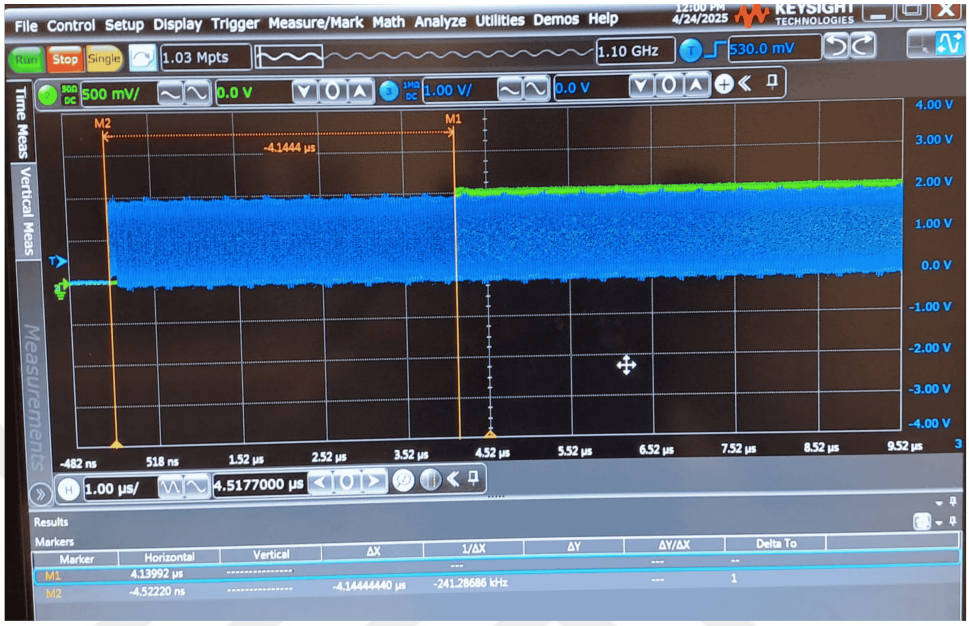


Figure 4.11: Observed lock time of the `MMCM` upon reactivation.

of approximately 240 ps, this was expected. The entire system was implemented digitally, with the filtered signal routed physically through external FPGA pins and looped back in. Interferences from both internal electrical deviations of the FPGA resources and the physical loop-back wiring contributed to the jitter.

In another test, a 180° phase shift was applied to the running clock signal, and the system's response time was observed (Fig. 4.13). Due to additional buffering inside the fractional clock divider IP and the physical loop-back delay, the phase



Figure 4.12: Evaluation of the filtered RF carrier waveform using an oscilloscope.

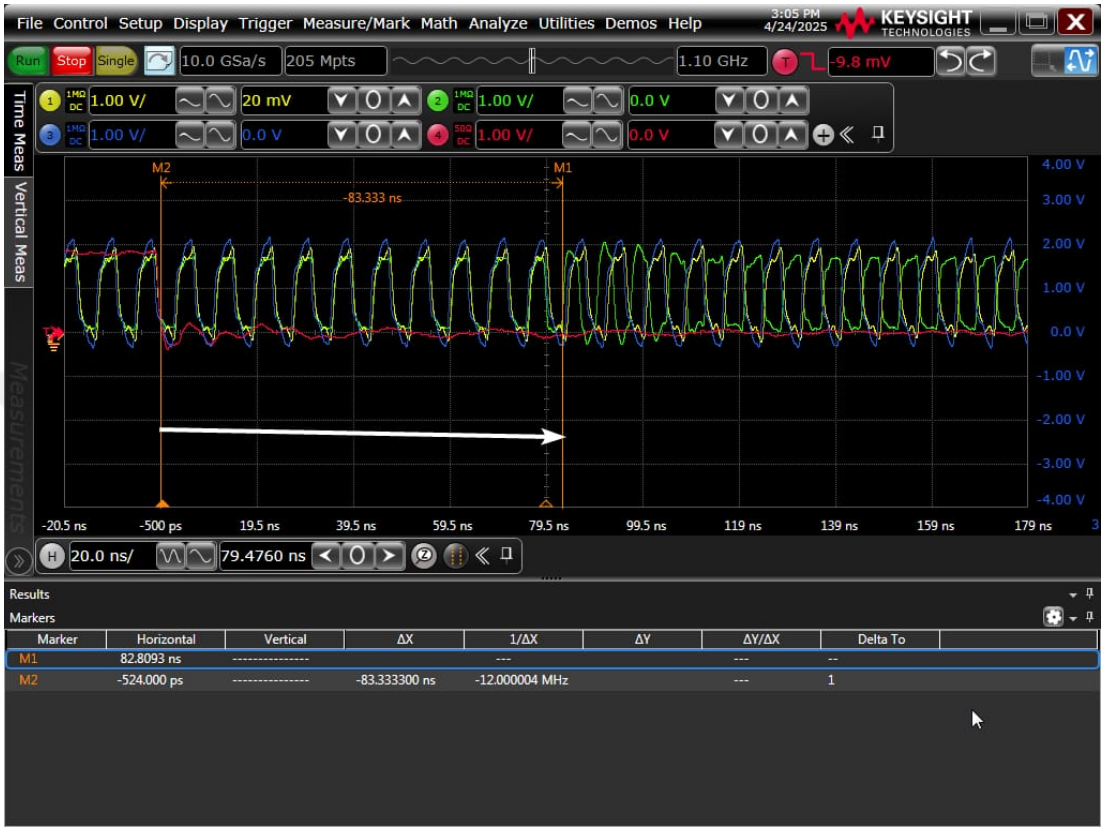


Figure 4.13: Observed delay for a 180° phase shift to take effect.

shift appeared only after roughly 80 ns (about eight clock cycles), which is almost negligible for the excitation pulse. Furthermore, in a digital system, where both the envelope and the carrier are synthesized together, this fixed delay time can be very easily compensated.

As for the last test before the MR experiment, the envelope PWM signals were validated. A sinc pulse shape with a time bandwidth product of 4 was generated, and the signal was observed over the oscilloscope (Fig. 3.8).

MR scans were performed to evaluate the performance of the RF pipeline in an MRI setting. The built-in RF system of the MRI was disabled and replaced by a custom RF amplifier developed by Arslan [16]. In these experiments, the RF controller was used only to supply an RF clock at the Larmor frequency. Since no RF spoiling was implemented, the TR of the imaging sequence was increased to values near or above the natural relaxation time of the phantom to avoid residual

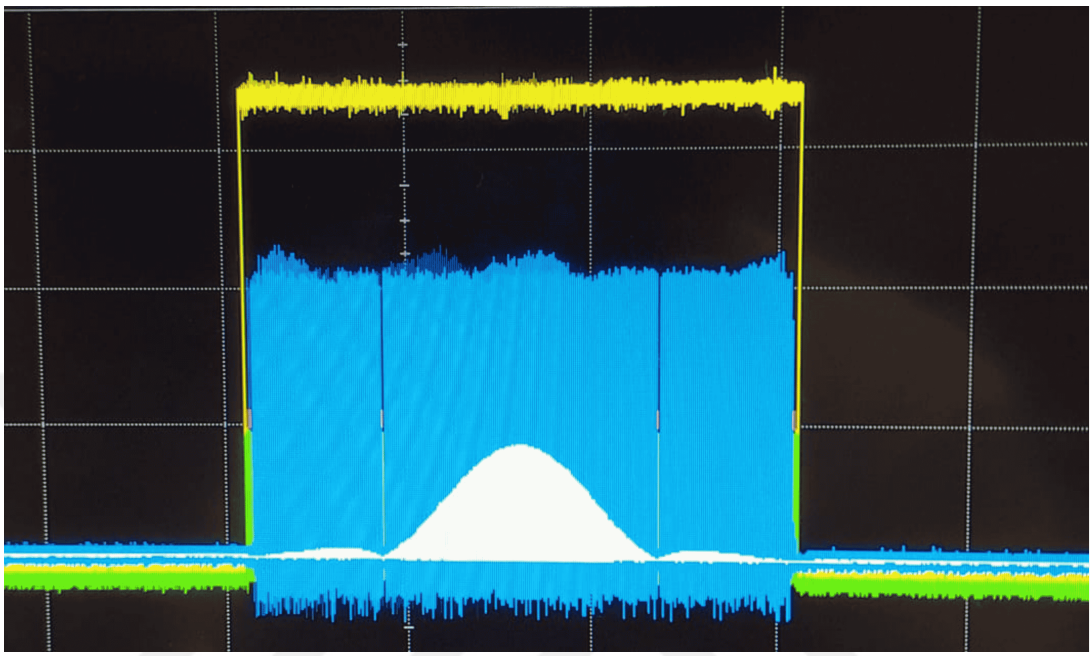


Figure 4.14: Example of a sinc-shaped RF envelope generated using PWM

excitation effects.

In the first scan (Fig. 4.15.a), the RF controller was configured to operate continuously. As expected, the persistent carrier leaked into the readout window and produced a zipper artifact. This artifact was intentionally used to probe the frequency accuracy of the generated carrier. Despite the high timing error margin observed via oscilloscope, the resulting artifact was limited to a thin frequency band and did not corrupt the overall image. The artifact converged to a narrow spatial region, suggesting that the frequency error was well-contained within the per-voxel bandwidth of the readout, which was approximately 10 kHz.

In the second scan (Fig. 4.15.b), the RF controller was only activated during RF transmit events. Although the `MMCM` locking delay was still present, it did not visibly impact the excitation pulse. As a result, the zipper artifact was successfully eliminated.

With the completion of the MR experiments, we presented all the results. Due to time constraints, the RF controller subsystem could not be explored in full

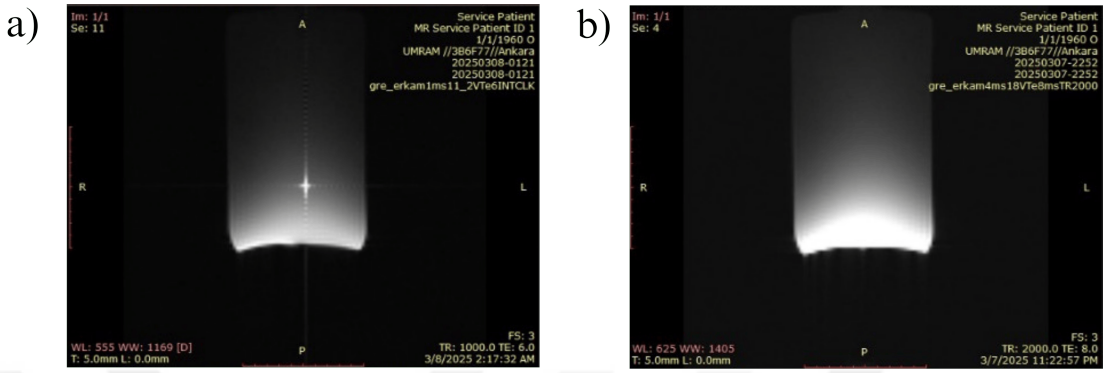


Figure 4.15: MR images acquired during RF subsystem validation. a) RF controller operated continuously, revealing a zipper artifact. b) The RF controller is activated only during RF events, eliminating artifacts.

detail. Several advanced features—such as high-resolution phase control and dynamic phase shifting for RF transmit arrays—were developed but ultimately left untested. Moreover, the implemented system was far from ideal, with significant jitter and system-level delays.

Despite these limitations, we had some alternative approaches that could serve as a solution. We briefly investigated different on-chip methodologies, such as time-to-digital converters and Vernier delay lines using the `CARRY8` primitive [51], to enhance the timing resolution. Although many proposed techniques could not support runtime adjustments on the clock, there were also several promising articles, such as the work of Wang et al. [52]. The appendix explains one strategy to resolve the jitter problem (see Appendix B.1).

Chapter 5

Pulse Sequence Design for Gradient Arrays

Pulse sequence design for gradient arrays remains a rather unexplored field in the literature. Unlike conventional pulse sequences, gradient arrays can accommodate numerous gradient channels of arbitrary positions. While providing a much higher degree of freedom from a pulse design perspective, the overall complexity of the hardware system also increases. This complexity poses problems, especially for oblique imaging, where hardware constraints are usually conserved for conventional systems, as explained in Chapter 2.2.2. The maximum slew rates can be de-rated up to $\sqrt{3}$ in conventional systems, let alone an n-channel array system where, in the worst case, the overall system can be forced to be de-rated up to $\sqrt{n}$. There have been different approaches to tackle the inefficiency, but all were conservative and limited the potential slew rates. As a solution, Atalar and McVeigh [26] had proposed a new rotation algorithm to ensure the system runs in peak condition. However, since the algorithm fixes the whole sequence onto one oblique plane, it was not applicable to the MR consoles, which need to be flexible and fast when using different rotation angles, such as the algorithm of Bernstein and Licato [27].

That being said, gradient array sequences have different requirements. As

each new oblique plane requires a new set of optimized current weight matrices, the whole sequence inevitably needs to be regenerated. Therefore, redesigning the sequences in the physical coordinate system [26] can be considered a golden standard for rotating array sequences, where any rotation angle can safely accommodate the original hardware limits of the system. Simultaneously, as the system hardware can be fully utilized, minimum scan durations that are not possible on non-oblique axes can be obtained.

When describing gradient array pulse sequences, one other challenge is finding an acceptable format to store and export the sequence file. The most naive approach would be to export the transient amplitude values of each individual channel. However, considering an array system of forty channels, amplifiers that work at a 10 kHz frequency, and a sequence that lasts one second, the resulting sequence will occupy a memory size of approximately 16 MB. Furthermore, the data size increases exponentially with time and the frequency of the amplifiers. Another method would be to compress the data using its first-order derivatives. If the derivative of the data is constant for more than two raster times, compression reduces the overall data size. However, this approach still fails to be the most efficient, as repeated waveforms, which are abundant in MRI pulse sequences, are explicitly expressed. In this regard, PulSeq [28], an open-source framework for developing MRI pulse sequences, is a great candidate to accommodate the complex nature of array sequences.

The following sections explain the envisioned pulse sequence design pipeline for the gradient array systems. To limit the scope of the pipeline for realizability, our system assumes different sets of optimized current weight matrices are precalculated for different events. We first expand the key concept of "designing in the physical coordinates" to propose an error-free rotation algorithm for array sequences to scan on oblique planes. Simultaneously, we show the benefits of designing in the physical coordinates by optimizing the system hardware by reducing the minimum possible TE and TR times. Lastly, leveraging the PulSeq toolbox, we propose a readable export file format to store the sequence data with minimal memory expenditure.

5.1 Designing Array Sequences on the Physical Coordinate System

Designing pulse sequences that can run on any oblique plane is a non-trivial problem, as previously mentioned in Chapter 2.2.2. This problem worsens for gradient arrays as multiple channels exist on arbitrary positions. Furthermore, hardware violations are not the only problem that worsens with the arrays. Eddy currents on the cryostat [49], or the risk of PNS in the body [50], pose complex problems to solve. Though these problems can be almost entirely avoided by optimizing a current weight matrix to scale individual gradient channels, optimally using pulse sequences further helps to mitigate the problems. As for the solution, we propose designing the pulse sequences on the physical coordinate system as done in the work of Atalar and McVeigh [26].

In the proposed method, the full potential of array sequences is not exploited due to their complexity. Theoretically, it is possible to design pulses that do not reflect the conventional description of slice selection, phase encoding, or readout. With arbitrary gradient coil positions, sequences that have additional properties, such as gradient shimming, can be realized. However, it is a big challenge to accommodate and optimize the wide range of applications.

Instead, we stay within the boundaries of conventional pulse design methodology, but extend it to linearly scaled multichannels using different sets of optimized current weight matrices. With this assumption, we can cover the whole gradient array sequence design pipeline. By designing in the physical coordinate system, we further solidify the pipeline in the sense that any oblique image plane can be realized without compromising the hardware system limits. However, before going into the pipeline, it is necessary to first go through an updated waveform optimization algorithm that Eq. 2.12 provides.

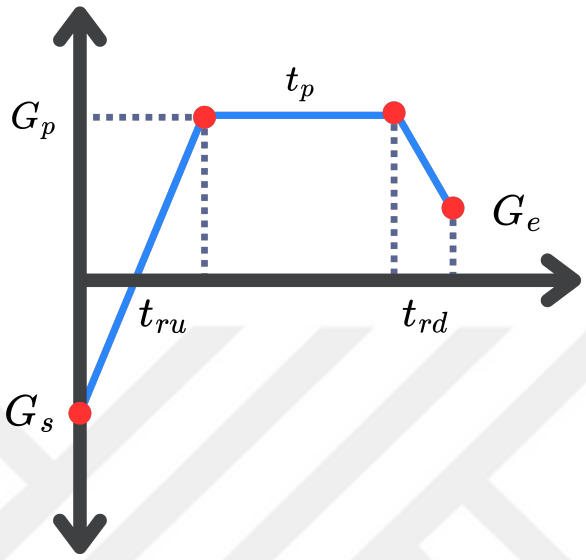


Figure 5.1: An example trapezoidal waveform

5.1.1 Waveform Optimization Algorithm

The waveform optimization algorithm is used to generate waveforms that can give the shortest duration under the constraints of starting amplitude G_s, ending amplitude G_e, and an area to cover a. Since it is known that trapezoidal waveforms with the maximum slew rate ramps generate the shortest possible duration to cover a particular area, we can use piecewise integration on an example trapezoid shown in Fig. 5.1 to find the corresponding relations, which are bound under the restrictions of 5.1:

$$|G_p| < G_{\max}, |SR| < SR_{\max} \tag{5.1}$$

where $G_{\max}$ and $SR_{\max}$ are the maximum gradient amplitude and slew rate that the system can produce.

First, we focus on the rising edge, and define a function $G(t)$ that starts with starting amplitude G_s and ends with the peak amplitude G_p during a rise time

of t_{ru} with the slope of α. We can then find $G(t)$ as 5.2:

$$G(t) = \alpha t + G_s$$

$$G(t_{\mathrm{ru}}) = \alpha t_{\mathrm{ru}} + G_s = G_p$$

$$G(0) = G_s$$

$$G(t) = \frac{G_p - G_s}{t_{\mathrm{ru}}} t + G_s \tag{5.2}$$

To find the area underneath, we can integrate from $t = 0$ to $t = t_{\mathrm{ru}}$, and obtain 5.3:

$$\int_0^{t_{\mathrm{ru}}} G(t)\, dt = \frac{G_s + G_p}{2} t_{\mathrm{ru}} \tag{5.3}$$

After repeating the same operation on the other edges, we can find a general solution for the area of the trapezoidal waveform as Eq. 5.4:

$$a = \frac{G_p + G_s}{2} t_{\mathrm{ru}} + G_p t_p + \frac{G_e + G_p}{2} t_{\mathrm{rd}} \tag{5.4}$$

where t_{ru}, t, t_{rd} are rise-up, peak, and rise-down times, respectively. With this analytical solution, finding the minimum duration under the system limits is trivial. However, in MRI systems, the gradient waveforms are not calculated as continuous signals but rather digitized at time intervals called gradient raster times. Consequently, the analytical solution changes to Eq. 5.5.

$$a = \left(\frac{G_s + G_p}{2} n_{\mathrm{ru}} + G_p n_p + \frac{G_e + G_p}{2} n_{\mathrm{rd}} \right) \Delta t \tag{5.5}$$

where n_{ru}, n_{p}, n_{rd} are integer numbers of raster times Δt it takes to rise up, peak, and rise down, respectively. As the current problem has more restrictions now, the convergence to the minimum duration is not always satisfied for trapezoidal waveforms. As an alternative solution, the corner points of the trapezoid are relaxed, and their amplitudes are not necessarily equal[1]. With the modification, the new quadrilateral waveform can be parametrized as shown in Fig. 5.2. As for

[1]The idea to have varying amplitudes on corner points was proposed by Prof. Dr. Maxim Zaitsev from Uniklinik Freiburg during a summer research internship.

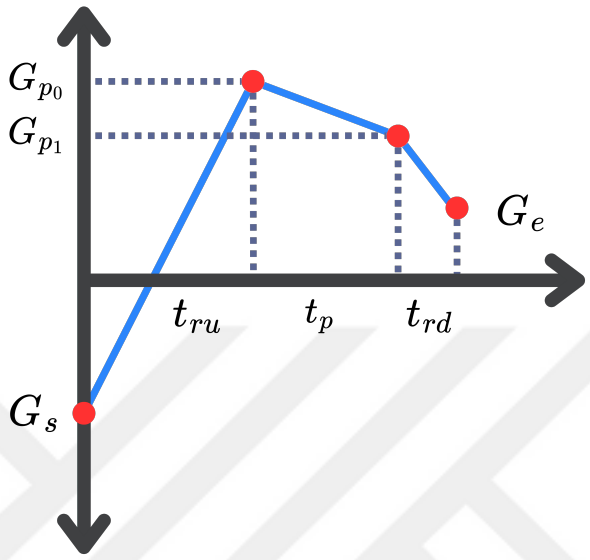


Figure 5.2: New quadrilateral waveform with adjustable peak corner points

the area equation shown in Eq. 5.5, we define two adjustable corner points of G_{p_0} and G_{p_1} and obtain the general solution of Eq. 5.6.

$$a = \left(\frac{G_s + G_{p_0}}{2} n_{\mathrm{ru}} + \frac{G_{p_0} + G_{p_1}}{2} n_{\mathrm{p}} + \frac{G_e + G_{p_1}}{2} n_{\mathrm{rd}} \right) \Delta t \tag{5.6}$$

Eq. 5.6 presents a complex problem with multiple dependent variables and constraints. Each time step is placed on raster times, maximum slopes are limited by maximum slew rates, and the maximum amplitudes are restrained by the maximum gradient strength. Under such conditions, one can set the slopes and peak gradient amplitudes to the maximum to obtain the shortest duration. However, the digitized time steps pose a significant challenge in this regard. Therefore, a search algorithm is needed to find the optimum solution.

The search used in this work fixes the shape duration to specific values to test the feasibility. If the current duration can indeed generate a shape that can accurately cover the desired area, it is shortened and tested again. Likewise, if the current duration cannot satisfy the area requirements, then the timing is relaxed. To improve the performance, a binary search algorithm described in Fig. 5.3 is

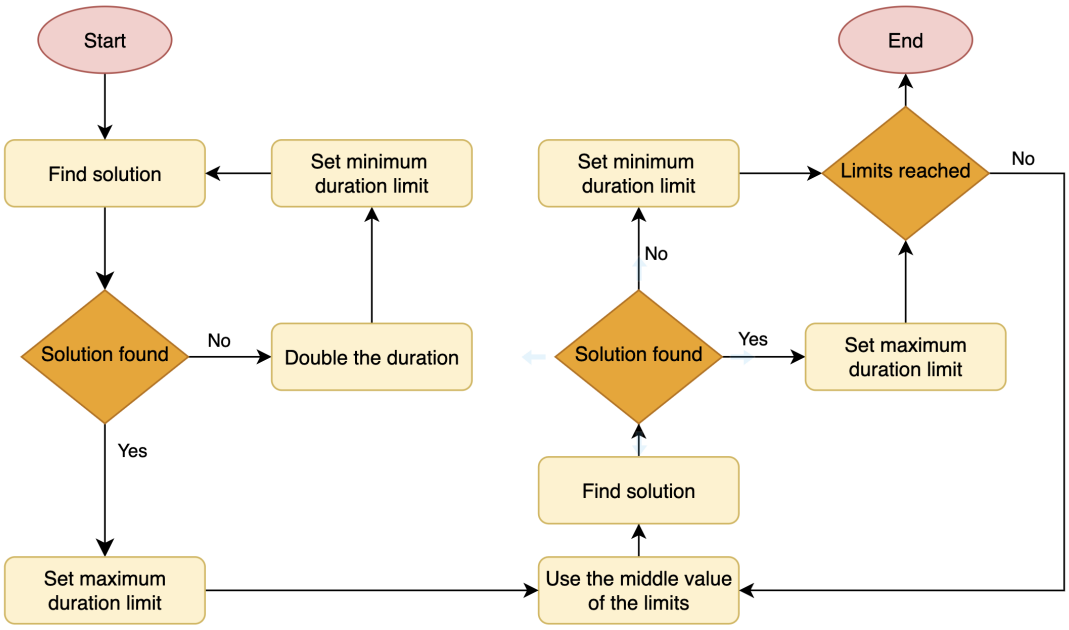


Figure 5.3: Binary search algorithm

applied for optimum computational speed.

The feasibility of each duration is conducted inside a solver function. The solver first checks if the solution can ever be found using the maximum slew rate and maximum amplitudes without considering the digitization. If Eq. 5.7 holds true, then since there is no potential solution, the solver function exits prematurely.

Early exit case:

$$G_p = G_{\max}, \qquad SR = SR_{\max}, \qquad t_{\mathrm{ru}}, t_p, t_{\mathrm{rd}} \neq n\Delta t,$$
$$t_{\mathrm{ru}} = \frac{|G_s - G_{max}|}{SR_{max}}, \quad t_{\mathrm{rd}} = \frac{|G_e - G_{max}|}{SR_{\max}}, \quad t_p = Dur - t_{ru} - t_{rd}$$

$$a > \frac{G_s + G_{max}}{2} t_{\mathrm{ru}} + G_{max} t_{\mathrm{p}} + \frac{G_e + G_{max}}{2} t_{\mathrm{rd}} \tag{5.7}$$

If Eq. 5.7 is false, then the solver checks if it can find an easy solution where

$t_p > 0$. First, it finds the minimum rise-up and rise-down times that can reach the maximum gradient strength. Then, the durations are adjusted to fall below the maximum slew rate using the ceiling operator alongside the gradient raster time for digitization. Lastly, the peak amplitude G_p is found analytically according to the desired area value Eq. 5.9.

Easy solution with $t_p > 0$:

$$G_p \neq G_{\max}, \qquad SR \neq SR_{\max}, \qquad t_{\mathrm{ru}}, t_p, t_{\mathrm{rd}} = n\Delta t,$$

$$t_{ru_{max}} = \left\lceil \frac{|G_s - G_{max}|}{SR_{max}} \right\rceil \Delta t, \quad t_{rd_{max}} = \left\lceil \frac{|G_e - G_{max}|}{SR_{max}} \right\rceil \Delta t$$
$$t_{p_{min}} = Dur - t_{ru_{max}} - t_{rd_{max}} \tag{5.8}$$

$$G_p = -\frac{t_{ru} G_s + t_{rd} Ge - 2a}{t_{ru} + t_{rd} + 2t_p} \tag{5.9}$$

After finding the analytical solution for the G_p from Eq. 5.9, the solver checks the new slopes with the new G_p. If the slew rates are in bounds, the solution is found and the resulting times and amplitudes are returned. If not, the solver proceeds to the in-depth search where the corner points of G_{p_0} and G_{p_1} are adjustable.

The solver first tries to limit the search range for the in-depth search. As Eq. 5.7 certifies the existence of maximum values for t_{ru} and t_{rd}, it searches for a minimum value to start the search.

Minimum t_{ru} and t_{rd} values to start the search:

$$G_p \neq G_{\max}, \qquad SR \neq SR_{\max}, \qquad t_{\mathrm{ru}}, t_p, t_{\mathrm{rd}} = n\Delta t,$$

$$t_{\mathrm{ru}} = \left\lfloor \frac{|G_s - G_{max}|}{SR_{max}} \right\rfloor \Delta t, \quad t_{\mathrm{rd}} = \left\lfloor \frac{|G_e - G_{max}|}{SR_{max}} \right\rfloor \Delta t, \quad t_p = Dur - t_{ru} - t_{rd}$$

For G_p value found by Eq. 5.9, repeat until:

$$\text{if:} \quad a > \frac{G_s + G_p}{2} t_{\text{ru}} + G_p t_{\text{p}} + \frac{G_e + G_p}{2} t_{\text{rd}}$$
$$\text{else:} \quad G_p = \frac{G_p}{2}$$

Although dividing G_p by two results in a rather conservative limit, it still limits the search space and finds the minimum times of $t_{ru_{min}}$ and $t_{rd_{min}}$.

$$t_{ru_{min}} = \left\lfloor \frac{|G_s - G_p|}{SR_{max}} \right\rfloor \Delta t, \quad t_{ru_{min}} = \left\lfloor \frac{|G_e - G_p|}{SR_{max}} \right\rfloor \Delta t,$$
$$t_{p_{max}} = Dur - t_{ru_{min}} - t_{rd_{min}} \tag{5.10}$$

At this point, the next goal is to slowly increment the time steps of n_{ru} and n_{rd} one step at a time to reach a point where $a_{calc} > a_{target}$. The core idea of this step is that if the target equation can be satisfied with the maximum slew rates, as the corner points of G_{p_0} and G_{p_1} are highly adjustable, we can find a corresponding solution within the system limits.

Increments:

$$n_{ru} = n_{ru} + 1, \qquad n_{rd} = n_{rd} + 1$$
$$G_{p_0} = G_s + n_{ru} \Delta t SR_{max}, \qquad G_{p_1} = G_e + n_{rd} \Delta t SR_{max}$$

Target:

$$a < \left(\frac{G_s + G_{p_0}}{2} n_{\text{ru}} + \frac{G_{p_0} + G_{p_1}}{2} n_{\text{p}} + \frac{G_e + G_{p_1}}{2} n_{\text{rd}} \right) \Delta t \tag{5.11}$$

Once Eq. 5.11 is true, the solver proceeds to find the analytical solutions for G_{p_0} and G_{p_1}.

$$t_{\text{ru}} = \frac{|G_s - G_p|}{SR_{max}} \Delta t, \quad t_{\text{rd}} = \frac{|G_e - G_p|}{SR_{max}} \Delta t, \quad t_p = Dur - t_{ru} - t_{rd}$$

if $G_{p_0} < G_{p_1}$, fix G_{p_1} and find G_{p_0} as in Eq. 5.12.

$$G_{p_0} = -\frac{t_{ru} G_s + t_{rd} Ge + G_{p_1}(t_p + t_{rd}) - 2a}{t_{ru} + t_p} \tag{5.12}$$

if $G_{p_0} > G_{p_1}$, fix G_{p_0} and find G_{p_1} as in Eq. 5.13.

$$G_{p_1} = -\frac{t_{ru}G_s + t_{rd}Ge + G_{p_0}(t_p + t_{ru}) - 2a}{t_{rd} + t_p} \tag{5.13}$$

The solver exits with the solution if the final amplitude and time values satisfy the slew rate constraints. Otherwise, the solver conducts a linear search by trying every time point on raster time steps in the previously found minimum Eq. 5.10 and maximum Eq. 5.8 limits. The solver returns the solution if it can find one. Otherwise, it returns a null value.

The process explained above covers the main pipeline of the waveform optimization algorithm. However, some cases in the search algorithm lead to failure. The resulting amplitude values can breach the system limits as the solver increments the rise-up and rise-down times. In these edge cases, the solver adjusts itself accordingly.

Edge case 1, $G_{p_0} > G_{max}$:

At first glance, equating $G_{p_0} = G_{max}$ may seem to be an obvious choice. Surprisingly, it is not. There are cases where it is better to take a step back and equate $n_{ru} = n_{ru_{max}} - 1$. Therefore, the solver first compares the corresponding area values of $a_{n_{ru_{max}}} < a_{(n_{ru_{max}}-1)}$, then chooses the time step according to Eq. 5.14.

$$G_{p_0} = G_{max}, \quad G'_{p_0} = (n_{ru_{max}} - 1)SR_{max}\Delta t + G_s$$

$$\frac{G_{max} + G_s}{2}t_{ru_{max}} + \frac{G_{max} + G_{p_1}}{2}t_p > \frac{G'_{p_0} + G_s}{2}(n_{ru_{max}} - 1)\Delta t + \frac{G'_{p_0} + G_{p_1}}{2}(n_p + 1)\Delta t$$

$$G'_{p_0} > \frac{G_{max}(t_{ru} + t_p) + G_s - G_{p_1}}{t_{ru_{max}} + t_p} \tag{5.14}$$

Once this condition is fulfilled, the increments for the rise-up time are fixed, and only the rise-down time is incremented. If G_{p_1} is the one that breaches the hardware limit, the equations can be rewritten to accommodate it.

Edge case 2, $\frac{|G_{p_0} - G_{p_1}|}{t_p} > SR_{max}$:

Sometimes the slope between the corner points can violate the system limits. In such a case, the corner with greater amplitude is decreased to the system limit. If we assume $G_{p_0} > G_{p_1}$:

$$G_{p_0} = G_{p_1} + t_p SR_{max}$$

Edge case 3, $t_p = 0$:

As the last edge case, the solver covers the triangular waveforms. It performs a linear search for time points that lie under the minimum Eq. 5.10 and maximum Eq. 5.8 limits. This case is also an exit condition, as there are no more points to search.

With the edge cases covered, the solver function is complete. It can generate the waveforms that cover a particular area from arbitrary starting and ending amplitudes with the shortest duration.

As an extension of the solver function, since the minimum allowable duration for the input parameters is known, any other waveform with larger durations can be generated.

5.1.2 Error-Free Rotation for Gradient Array Sequences

Unlike conventional sequences of three major axes, gradient array sequences can accommodate any number of channels in arbitrary positions. While providing a much higher degree of freedom, the multichannel nature of arrays makes it hard to develop sequences, let alone use existing rotation algorithms for oblique imaging.

It has been previously demonstrated by [11, 13] that a hybrid of conventional and gradient array systems can be used to image. However, these studies only use axial slices, without the need to rotate. Therefore, it is a rather novel issue to address. The first work on oblique array sequences was conducted as a conference

abstract in 2024 and was presented as a digital poster [53].

The previous subsection describes the primary tool that allows efficient generation of waveforms. However, that tool alone is not enough to satisfy the requirements of error-free rotation in array sequences. It needs to be fit into the concept of designing pulse sequences in physical coordinates rather than the logical coordinates, proposed by Atalar and McVeigh [26].

The sequence design pipeline starts by designing a sequence in the logical coordinate system. As briefly explained in Chapter 2.2.1, fixed events, such as RF and readout, have predetermined shapes with gradient amplitudes, durations, and area values. The input parameters almost always fix these waveforms and should not be changed by the algorithm. The oblique array sequence design, thus, first calculates the parameters on the logical coordinate system, then maps onto the physical coordinate of the oblique imaging plane using Eq. 2.12. It abides by the durations set by the input parameters, such as the number of sampling points and receiver bandwidth, and fills the RF and readout events.

The next step is to design the dead-times. As aforementioned, dead periods are where the k-space trajectory is repositioned. Thus, the most important factor is the area underneath. The overall shape of the waveform can be manipulated to some extent. Therefore, after mapping the area previously calculated in the logical coordinates to the physical coordinates using Eq. 2.12, all the parameters required for the waveform optimization algorithm are set, and the dead period waveforms are generated.

In this step, it is also possible to adapt different optimization targets to the dead period waveforms, alongside the optimized current weight matrix, to fine-tune the sequence. Some examples can be seen in the GrOpt toolbox [54], where waveforms that perform eddy current compensation, PNS optimization, or dead-time minimization are generated. Similarly, since the sequence is already reformed in the physical coordinate system, it becomes possible to apply additional gradient waveform optimization schemes on any oblique imaging plane. However, special care must be considered, as the optimization targets increase with the number

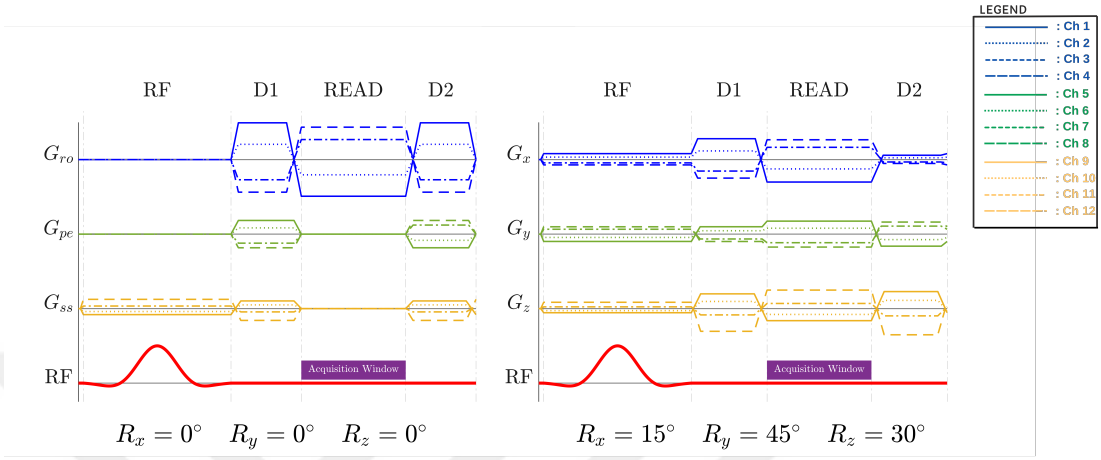


Figure 5.4: An example 12-channel bSSFP GRE gradient array sequence designed in the logical coordinates and its rotated and regenerated counterpart in the physical coordinate system

of channels. Different channels can either act destructively or additively on the generated fields. For this work, this aspect was not explored in depth. We only used the waveform optimization tool to generate the dead-time gradient waveforms under a specific time constraint.

As the final step, the previously calculated set of optimized current weight matrices is applied, and the gradients are scaled to generate the sequence for gradient array systems. The current weight matrices have scaling factors in the range of $[-1, 1]$; therefore, the risk of violating hardware system limits vanishes completely. After repeating the process for every other TR with the corresponding mapped area and amplitude values, the sequences for the gradient arrays can be generated. An example bSSFP GRE sequence for gradient arrays with four channels on each major axis can be seen in Fig. 5.4.

Since the sequence is regenerated in the physical domain, the previous timing constraints of the waveform shapes or sequence parameters, such as TE and TR values, are preserved without any issues. In fact, the timing constraints are relaxed in certain oblique planes, and shorter scan times can be obtained, as was shown by Atalar and McVeigh [26], and will be shown with larger improvements

in the next section of Chapter 5.2.

5.2 Dead-Period Minimization Algorithm

Minimization of dead periods was originally the work of Atalar and McVeigh [26]. The main idea was to optimize the dead times by using the maximum slew rate possible in the trapezoidal waveforms, thereby obtaining hardware-optimized trapezoids. We extend the same methodology to the arrays.

Unlike the conventional sequences, when transformed to an oblique plane, gradient array sequences have greater potential to reduce the dead times for shorter TE and TR values due to their multichannel nature. However, obtaining the minimum possible dead time, which must stay within the boundaries of the hardware limits, is not trivial.

Due to the phase encoding steps, the area of the phase encoding gradient changes in each TR in the pulse sequences. This complicates the process since the time values for each TR should be constant throughout the whole sequence. Therefore, finding a suitable time interval that accommodates each mapped channel waveform throughout the whole sequence becomes a challenge. In this context, the primary goal shifts to finding the maximum duration among the array of possible hardware-optimized trapezoids.

Suppose the sequence remains on an axial axis, since the starting and ending amplitudes remain constant. In that case, it is possible to simply use the minimum duration of the maximum area of the phase encoding gradients. However, when mapped on an oblique plane, the pure phase encoding gradient distributes onto each of the three major axes of x, y, and z. Without any control on the distribution of the area values, the possible minimum duration for dead periods varies in each TR on each channel. In this case, one can try to generate the whole array sequence, find the maximum duration among different optimized dead period waveforms without considering the constant duration requirement, and select

it as the constant dead time. However, this brute force method is highly inefficient and will take a very long time to generate a sequence, especially for the gradient arrays. Instead, testing with only three waveforms per dead period is enough.

Before describing the algorithm, there are four major points to consider:

1. In the pulse sequence, assuming a Cartesian trajectory, RF and readout gradients have constant amplitudes throughout the whole sequence.
2. Dead periods reside between RF and readout events. Their starting and ending amplitudes are equal to either the slice select or the readout gradient amplitude. Therefore, these amplitudes never change for individual dead periods.
3. In the pulse sequence, the area values for each phase encoding step can be calculated using a simple matrix multiplication. The largest area values for each of the three major axes of x, y, and z individually provide the most demanding phase encoding step.
4. Optimized current weight matrix values are bound in the range of $[-1, 1]$. Using the absolute maximum current weight for each individual, the three major axes allow finding the most demanding channel.

Therefore, for example, if the minimum duration of the waveform with the largest weight and the largest area value for the x-axis is found, the value will hold for all other channels in the x-axis. In other words, it is enough to find the maximum absolute weight among the set of optimized current weight matrices, and use it to scale the most demanding phase encoding steps from$x$, y, and z axes. As a result, the search for the maximum duration among optimized minima is reduced to three waveforms per dead period. The process is described in the flow diagram presented in Fig. 5.5.

The rest of the procedure is relatively straightforward after finding the minimum possible dead period duration that will not cause any hardware system

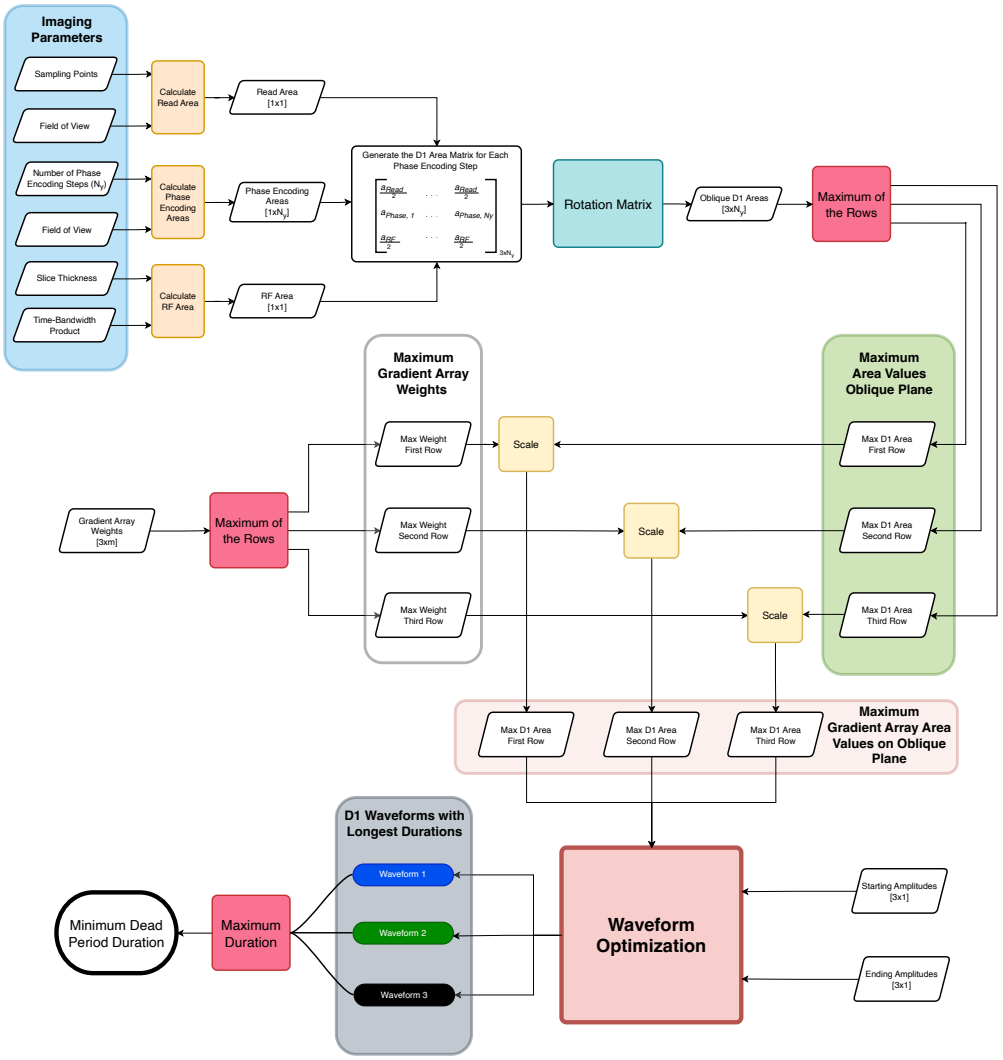


Figure 5.5: Flow diagram explaining how to acquire the minimum dead-period duration

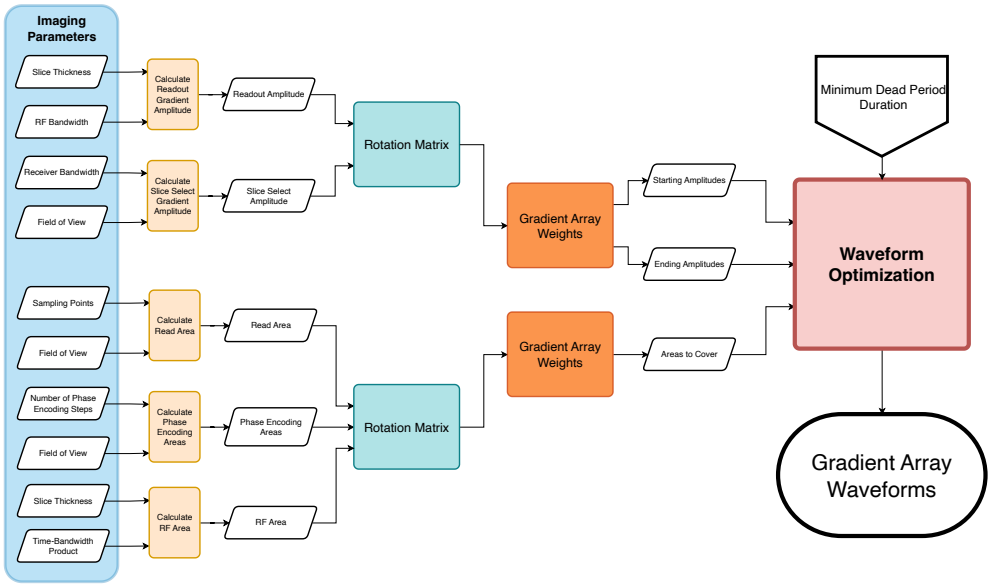


Figure 5.6: Flow diagram explaining how to generate the gradient array waveforms. The minimum dead period duration can be calculated by the process explained in Fig. 5.5

violations. The slice select and readout waveforms have already been generated. The only remaining waveforms to consider are the dead period waveforms. After imposing a timing constraint on the dead period waveforms, we can obtain the rest of the sequence as shown in the flow diagram Fig. 5.6.

5.2.1 Results and Comparison of Dead Times with the Conventional Sequences

In our simulations, we used current weights from a 288-channel body coil, optimized for increasing the PNS thresholds from the previous work of Babaloo [50]. We designed an example bSSFP GRE pulse sequence for both conventional and array coils. The imaging parameters were assumed as maximum gradient of 40 mT/m, maximum slew rate of 200 T/m/s, 360 mm FOV, matrix size 192×144,

Rotation Angles			Conventional Coil				Gradient Array Coil			
X	Y	Z	D1 (μs)	D2 (μs)	TE (ms)	TR (ms)	D1 (μs)	D2 (μs)	TE (ms)	TR (ms)
0	0	0	66	66	0.98	1.95	66	66	0.98	1.95
0	10	45	71	71	1.03	2.05	54	54	0.86	1.71
5	12	45	73	73	1.05	2.09	55	55	0.87	1.73
30	45	30	79	80	1.11	2.22	61	61	0.93	1.85
4	45	45	80	80	1.12	2.23	61	62	0.94	1.87

Figure 5.7: Comparison of the minimum possible D1, D2, TE, and TR values for conventional and array coils on five different oblique planes

8-mm slice thickness, 70° flip angle, 3 kHz/Pixel receiver bandwidth. In both cases, the imaging plane was flipped for 45° on three major axes individually ($46^3 = 97336$ samples). For each individual oblique plane, the respective dead-period minimization algorithm was applied.

Fig. 5.7 illustrates the dead-times (D1 and D2), TE and TR parameters of the bSSFP sequence in five distinct imaging planes for two cases:

1. Optimized for a conventional coil using the algorithm proposed by [26],
2. Optimized for the gradient array using our proposed method.

As a result, depending on the scan plane orientation, the array coils can reduce the dead times up to 25% compared to conventional coils, which are already hardware optimized and have their maximum hardware limits preserved. Furthermore, the TE and TR values can also be reduced up to 17%, which is a significant improvement.

5.3 Export File Format for Array Sequences

Unlike the everyday use of PulSeq [28] for conventional sequences, there exists another object called "External Gradients". External gradients, as gradient objects, are stored inside the sequence object. The distinction between external gradients and conventional gradients is established through their channel names. While the conventional gradients have the channel names of "x", "y", and "z", external gradients have integer numbers such as "1", "2"... in their descriptions. This allows a simple but effective convention, allowing hybrid MRI + array systems to be run together. Furthermore, different system limits can be provided to conventional and external gradients for better integration.

All being said, the external gradient support of PulSeq v1.5.0 [28] is still very native to the developer. In the aforementioned version, although the object definitions are ready, the number of external channels is hard-coded to 12, the plotting is conducted on the same pane as the conventional gradients of "x", "y", and "z", and the export file only prints an additional file that accommodates the transient gradient amplitudes of each channel in a concatenated format. All these limit the potential of PulSeq for gradient arrays in the current state.

As an alternative,

1. An arbitrary number of channels should be supported
2. The plots should be migrated to another pane, where instead of "x", "y", and "z" directions, the waveforms should be enumerated according to their channel numbers
3. The output files should have a similar format to the conventional sequence outputs that are fed to the scanners.

Accommodating an arbitrary number of channels is a rather trivial topic and can be handled by modifying the sequence object parameters. Likewise, plotting the external channels on "x", "y", and "z" gradients is not the most suitable

approach, but it can still provide sufficient information. However, the output file format is tough to read, inefficient in file size, and unsuitable for different gradient array systems. Our proposal of an acceptable export file format specifically targets this aspect of PulSeq.

As explained in Chapter 2.2.3, the building block of PulSeq sequences is blocks. Blocks define different time intervals, making it very convenient to accurately accommodate various events. In the regular export format, blocks present the mainstream of sequential events for specific time intervals. Inside the blocks, any defined event maps to its corresponding object space, where more detailed descriptions can be found. Most importantly, repeated events can be mapped to the same description, which significantly reduces the file size, especially for MRI sequences where repeated waveforms are abundant.

We propose using the same approach for the external gradients, on a separate export file. Blocks should accommodate all gradient channels. Gradient objects should have the corresponding information of amplitudes, delays, shape, and time vectors. Separating the shape and time vectors from the gradient description makes it possible to map different gradient events to the same shape object and allows the interpreter to scale the relevant waveform with the corresponding amplitude information. To visualize the concept, an example sequence from Fig. 5.4 is reconstructed as a sequence file.

In the sequence file shown in Fig. 5.8, the gradient array system is assumed to be 12 channels. As shown in Fig. 5.4, channels from 1-4 represent the x-direction, channels from 5-8 represent the y-direction, and the channels from 9-12 represent the z-direction. One full TR consists of four blocks, and overall, four TRs are used to demonstrate the notation of gradient and trapezoid objects. Each of the four blocks from a single TR represents the RF, D1, READ, and D2 periods as defined in Fig. 5.4. The snippet from the export file in Fig. 5.8 shows the blocks, which define different gradient events for different numbers. The detailed description is as follows:

- **BLOCK 1 - RF:** Slice selection gradients on the z-axis are applied as

```
[DEFINITIONS]
AdcRasterTime 1e-07
BlockDurationRaster 1e-05
GradientRasterTime 1e-06
ExtGradNum 12

# Format of blocks:
# NUM DUR  G1  G2  G3  G4  G5  G6  G7  G8  G9  G10  G11  G12

  1 300    0   0   0   0   0   0   0   0   1   2   3   4
  2  66    5   6   7   8   9  10  11  12  13  14  15  16
  3 200   17  18  19  20   0   0   0   0   0   0   0   0
  4  66    5   6   7   8  21  22  23  24  13  14  15  16
  5 300    0   0   0   0   0   0   0   0   1   2   3   4
  6  66    5   6   7   8  25  26  27  28  13  14  15  16
  7 200   17  18  19  20   0   0   0   0   0   0   0   0
  8  66    5   6   7   8  29  30  31  32  13  14  15  16
  9 300    0   0   0   0   0   0   0   0   1   2   3   4
 10  66    5   6   7   8  33  34  35  36  13  14  15  16
 11 200   17  18  19  20   0   0   0   0   0   0   0   0
 12  66    5   6   7   8  37  38  39  40  13  14  15  16
 13 300    0   0   0   0   0   0   0   0   1   2   3   4
 14  66    5   6   7   8  41  42  43  44  13  14  15  16
 15 200   17  18  19  20   0   0   0   0   0   0   0   0
 16  66    5   6   7   8  45  46  47  48  13  14  15  16
```

Figure 5.8: Described Sequence Formatting - `[BLOCKS]`

`[TRAP]` events (Fig. 5.9). Since the same gradients are repeated on blocks 5, 9, and 13, it is possible to see the same event pointers on channels 9-12.

- **BLOCK 2 - D1:** Phase encoding gradients are applied in all directions. While y and z-directions accommodate [TRAP] events, due to non-zero ending amplitude, gradients in the x-direction have `[GRAD]` events. Due to their repeated nature, events on the x and z-axes are represented with the same numbers on blocks 6, 8, 10, 12, 14, and 16.

- **BLOCK 3 - READ:** Readout gradients on the x-axis are applied as `[GRAD]` events. Repeated on blocks 7, 11, 15.

- **BLOCK 4 - D2:** Refocusing gradients are applied. The gradient events point to the same descriptions since the same waveforms from the D1 period are reused for the x and z directions. New descriptions are added only for phase encoding gradients on the y-axis.

As shown in Fig. 5.9, the denotations shown in 5.8 map to their corresponding gradient events. `[GRAD]` events describe the scaling amplitude, delay, and waveform shape with the corresponding time vector. The waveform amplitude

```
# Format of arbitrary gradients:
# id amplitude amp_shape_id time_shape_id delay
# ..      A       ..          ..          us
[GRADIENTS]
5  40   1   2   0
6  20   1   2   0
7 -20   1   2   0
8 -40   1   2   0
17  40   3   4   0
18  20   3   4   0
19 -20   3   4   0
20 -40   3   4   0

# Format of trapezoid gradients:
# id amplitude rise flat fall delay
# ..      A   us   us   us    us
[TRAP]
1    40 200 2500 200 100
2    20 200 2500 200 100
3   -20 200 2500 200 100
4   -40 200 2500 200 100
9    30 160 340 160   0
10    19 160 340 160   0
11   -15 160 340 160   0
12   -25 160 340 160   0
13    20 160 340 160   0
```

Figure 5.9: Described Sequence Formatting - [GRAD] & [TRAP]

```
# Sequence Shapes
[SHAPES]

shape_id 1
num_samples 4
0
-1
-1
1

shape_id 2
num_samples 4
0
160
500
660
```

Figure 5.10: Described Sequence Formatting - [SHAPES]

and time values then map to another object called [SHAPES] (Fig. 5.10). Unlike [GRAD], [TRAP] objects are a special type; they do not require additional shape description. Those special waveforms can be generated just using the provided information, as explained in Chapter 2.2.3.1.3.2.

Lastly, Fig. 5.10 shows the [SHAPES] object. As mentioned, the transient amplitudes and their corresponding time points are stored here. One crucial aspect of these shapes is their compressibility. If the described object has a fixed first derivative for more than four time points, PulSeq automatically applies its compression function to reduce the data size. The compression can significantly decrease the data size, especially for the arbitrary gradients defined in PulSeq.

In addition to the aforementioned objects, the rest of `[RF]`, `[ADC]`, or `[EXT]` objects can also be added to the formatting. However, it might be redundant if the gradient array system does not incorporate any RF, ADC, or trigger subsystems. The described methodology covers the most basic needs for a gradient array system, and can be readily implemented in the current PulSeq toolbox.

5.4 Chapter Summary

In this chapter, we addressed the pulse sequence design methodologies for gradient arrays and proposed a complete implementation pipeline. First, we expanded the concept of designing in the physical coordinates to propose an error-free rotation algorithm. Simultaneously, we presented the benefits of designing in the physical domain by minimizing the dead-periods to achieve the minimum possible TE and TR times that conventional systems cannot realize. Lastly, leveraging the PulSeq toolbox, we proposed a readable and efficient export file format to store the array sequences with minimal memory expenditure.

Chapter 6

Discussion and Conclusion

This thesis presented the design and implementation of spectrometer architectures for gradient array MRI systems, including a pulse sequence parser, gradient, and RF controllers. Throughout the process, we focused on building scalable, modular, and FPGA-based frameworks that seamlessly integrate into a larger design, while offering room for further extension.

In Chapter 3, we developed modular blocks that can be individually validated or jointly operated. We first extended the PulSeq sequence format and implemented a parser that converts it into hardware-addressable memory structures. We generated two outputs: one for simulation and debugging, and the other for physical memory programming. This dual-mode design significantly accelerated the development cycle.

Then, in Chapter 3.2, we built the gradient controller subsystem around the assumption that each array coil contributes a directional field component. Conventional gradient trajectories were translated into current values for each array channel using a matrix projection model. This controller was then integrated with a custom 40-channel gradient driver system, where successful MR images were obtained using four of the channels.

The gradient controller subsystem design provided a highly scalable framework.

Apart from the current directional coil configuration, it supports channels of arbitrary position and counts. The only limiting factor for the scaling is the FPGA's physical pin count and logic resources. However, this can be bypassed by introducing one or more distributor modules that relay the waveform data at high speed, either in parallel or hierarchically.

Lastly, in Chapter 3.3, we developed an RF controller subsystem with a novel digital RF clock generation approach. Departing from conventional DDS or analog PLL systems, we explored a purely digital strategy built on the resources of a single FPGA. We formulated the theoretical basis and implemented an arbitrary clock generator using FPGA primitives such as `OSERDESE3` and `MMCM`. By handling everything within the same chip, we displayed an architecture that avoids synchronization mismatches, reduces board complexity, and gains full control over timing parameters such as raster resolution and modulation frequency.

However, due to time limitations, the system imperfections of the RF controller could not be solved. Still, an advanced technique for reducing phase noise is outlined, which uses additional bits from the fractional clock divider accumulator in Appendix B.1. Currently, only the MSB is used to toggle the output clock, but the remaining bits carry fine-grained phase information. By leveraging this, correcting the misaligned phase via programmable delays is possible. These delays could be implemented using multiple resources inside the FPGA fabric, such as the propagation path delays. Since propagation delays between logic elements are deterministic and immune to jitter, carefully designed delay chains can achieve picosecond-level timing resolution. However, selecting the specific pathway requires digital gate registers such as LUTs or MUXs; therefore, relying purely on propagation path delays for programmable delay generation is impossible.

To address this, we proposed a method involving Xilinx's `CARRY8` primitives, which offer low-skew paths and modular tap points. A chain of such primitives can serve as a fine-grained delay generator—similar in concept to a Tapped Delay Line Time-to-Digital Converter (TDL-TDC). While typical TDL-TDC designs lack runtime tunability, the recent work of Wang et al. [52] demonstrates how to change tap delays with high resolution during operation. As explained in

Appendix B.1, adopting such a method would allow us to extract and correct the lower-bit phase offsets encoded in the divider, ultimately reducing jitter and ensuring a stable RF clock.

That said, building such a system is non-trivial. The number of delay lines and control logic required increases significantly, and routing constraints must be carefully managed. Nevertheless, the potential benefits of low jitter, cost reduction, and complete FPGA-based control make it a compelling direction for future research.

It is also worth mentioning that while our RF controller provides a promising architecture, we have not fully explored its capabilities. We were unable to test multi-slice excitation or drive multiple transmit channels with dynamic phase control. Furthermore, scaling the RF subsystem requires additional synchronization mechanisms, such as clock recovery units, to maintain alignment across multiple outputs. These aspects were beyond the scope of this thesis, but remain viable future directions.

After covering the spectrometer frameworks, we moved on to the unexplored field of gradient array pulse sequences with Chapter 5.

Previously in Chapter 2.4.1, we pointed out the challenges in sequence design for gradient arrays and discussed how conventional assumptions fall short when transitioning to array-based architectures. One of the key observations was the requirement for electromagnetic simulation tools to generalize the pulse sequence design in arrays. However, the problem could be significantly simplified with the introduction of system-level assumptions, such as directionality in the gradient coils. Under this model, array sequences could be derived by projecting conventional trajectories onto array channels.

In Chapter 2.2.2, we had yet another critical observation, which was the necessity of redesigning sequences in the physical coordinate system to implement an error-free rotation algorithm. Conventional sequences already suffer from hardware performance loss when rotated into oblique orientations, with a potential

derating of the slew rates by up to a factor of $\sqrt{3}$. This issue becomes even more severe in array systems, where the number of channels introduces additional constraints, and the effective system limits may need to be derated by as much as $\sqrt{n}$.

In Chapter 5, we built upon these observations and proposed an error-free rotation algorithm for array sequences. We demonstrated that, by regenerating the sequences in the physical coordinate system, it is possible to avoid any restrictions that come along with the rotation operations and acquire shorter scan times below the achievable limits. The temporal alignment of rotated gradient waveforms is preserved, the dead periods are minimized for tighter packing of sequence events, and shorter TE and TR values are obtained.

Chapter 5 also provided auxiliary tools to ease the pulse sequence design for gradient arrays, including a waveform optimization algorithm and an export file format. The developed waveform optimization algorithm found the minimum possible duration of a waveform under the restrictions of a pulse sequence. Though an analytical solution exists for a trapezoidal waveform, the system's hardware limitations, along with the digitization of the target waveform, rendered the problem non-convex and required a binary search. The proposed export file format, on the other hand, addressed a crucial gap in adapting array-compatible sequences to external hardware. By extending the PulSeq file structure to include projection weights and channel-specific waveform data, the format minimized memory overhead and preserved human readability.

As a future goal, the discussed pulse sequence design concepts can be unified within a dedicated software platform. This unified gradient array pulse sequence designer can store the field responses of individual channels and use them to synthesize the desired linear gradient fields. Different imaging modalities—such as simultaneous multi-slice excitation—or optimization targets—such as PNS and eddy current optimization—can be selected for a given scan plane, and the corresponding optimized sequence can be automatically generated. The final sequence can then be exported using the proposed PulSeq-compatible format and routed

through the developed sequence parser to program the spectrometer memory directly. Such a tool would complete the full cycle—from field response to waveform deployment—within a single design environment.

Final Remarks

In conclusion, this thesis lays the groundwork for fully FPGA-based, modular spectrometer architectures for gradient array MRI systems. From sequence generation to waveform deployment, we proposed coherent frameworks that can seamlessly come together and scale up. While some modules are still under development, particularly in the RF chain, the presented architecture offers a strong foundation. With additional engineering effort, the system can be expanded into a powerful platform for a standalone gradient array MRI system.

Bibliography

[1] P. R. Harvey and E. Katznelson, "The modular gradient coil: a holistic approach to power efficient and high performance whole-body mri without peripheral nerve stimulation," *MAGMA*, vol. 9, pp. 152–155, 1999.

[2] P. R. Harvey, "The modular (twin) gradient coil—high resolution, high contrast, diffusion weighted epi at 1.0 tesla," *MAGMA*, vol. 8, pp. 43–47, 1999.

[3] D. L. Parker and J. R. Hadley, "Multiple-region gradient arrays for extended field of view, increased performance, and reduced nerve stimulation in magnetic resonance imaging," *Magnetic Resonance in Medicine*, vol. 56, pp. 1251–1260, 2006.

[4] S. Y. Lee and Z. H. Cho, "Localized volume selection technique using an additional radial gradient coil," *Magnetic Resonance in Medicine*, vol. 12, no. 1, pp. 56–63, 1989.

[5] R. Kimmlingen, M. Gebhardt, J. Schuster, M. Brand, F. Schmitt, and A. Haase, "Gradient system providing continuously variable field characteristics," *Magnetic Resonance in Medicine*, vol. 47, no. 4, pp. 800–808, 2002.

[6] P. R. Harvey and E. Katznelson, "Modular gradient coil: A new concept in high-performance whole-body gradient coil design," *Magnetic Resonance in Medicine*, vol. 42, pp. 561–570, Sept. 1999.

[7] A. Welz, C. Cocosco, A. Dewdney, D. Gallichan, F. Jia, H. Lehr, Z. Liu, H. Post, H. Schmidt, G. Schultz, F. Testud, H. Weber, W. Witschey, J. Korvink, J. Hennig, and M. Zaitsev, "Development and characterization of an

unshielded patloc gradient coil for human head imaging," *Concepts in Magnetic Resonance Part B: Magnetic Resonance Engineering*, vol. 43, pp. 111–125, 2013.

[8] J. Hennig, A. M. Welz, G. Schultz, J. Korvink, Z. Liu, O. Speck, and M. Zaitsev, "Parallel imaging in non-bijective, curvilinear magnetic field gradients: a concept study," *MAGMA*, vol. 21, no. 1-2, pp. 5–14, 2008.

[9] G. Schultz, P. Ullman, H. Lehr, A. M. Welz, J. Hennig, and M. Zaitsev, "Reconstruction of mri data encoded with arbitrarily shaped, curvilinear, non-bijective magnetic fields," *Magnetic Resonance in Medicine*, vol. 64, no. 5, pp. 1396–1409, 2010.

[10] D. Gallichan, C. A. Cocosco, A. Dewdney, G. Schultz, A. Welz, J. Hennig, and M. Zaitsev, "Simultaneously driven linear and nonlinear spatial encoding fields in mri," *Magnetic Resonance in Medicine*, vol. 65, no. 3, pp. 702–714, 2011.

[11] K. Ertan and E. Atalar, "Simultaneous use of linear and nonlinear gradients for b1+ inhomogeneity correction," *NMR in Biomedicine*, vol. 30, no. 9, 2017.

[12] K. Ertan, S. Taraghinia, A. Sadeghi, and E. Atalar, "A z-gradient array for simultaneous multi-slice excitation with a single-band rf pulse," *Magnetic Resonance in Medicine*, vol. 80, no. 1, pp. 400–412, 2017.

[13] S. Littin, *Development and Applications of a Highly Flexible Nonlinear Spatial Encoding System for Magnetic Resonance Imaging*. Phd thesis, Fakultät für Mathematik und Physik, Albert-Ludwigs-Universität, Freiburg im Breisgau, Sept. 2018.

[14] E. Aydın, R. Babaloo, and E. Atalar, "Thermal model in feedforward control of the gradient array system in mri," in *Proceedings of the 2023 ISMRM Annual Meeting*, (Toronto, Canada), International Society for Magnetic Resonance in Medicine, 2023.

[15] E. Aydın, M. E. Öztürk, and E. Atalar, "Scalable gradient array power amplifier design using digital current feedback," in *ISMRM Annual Meeting*, 2025.

[16] A. E. Arslan, "A 600w on-coil class-e rf power amplifier array with dynamic phase control for 3t mri," master's thesis, Bilkent University, Ankara, Turkey, Sept. 2022. Submitted to the Graduate School of Engineering and Science, Department of Electrical and Electronics Engineering.

[17] D. G. Nishimura, *Principles of Magnetic Resonance Imaging*. Stanford, CA: Stanford University, 2010. Online textbook.

[18] F. Bloch, "Nuclear induction," *Physical Review*, vol. 70, no. 7-8, pp. 460–474, 1946.

[19] M. Loecher, P. Magrath, E. Aliotta, and D. B. Ennis, "Time-optimized 4d phase contrast mri with real-time convex optimization of gradient waveforms and fast excitation methods," *Magnetic Resonance in Medicine*, vol. 82, no. 1, pp. 213–224, 2019.

[20] M. A. Bernstein, W. H. Perman, M. C. Besomi, and D. M. Thomasson, "Pulse sequence generated oblique magnetic resonance imaging: Applications to cardiac imaging," *Medical Physics*, vol. 13, no. 5, pp. 648–657, 1986.

[21] W. A. Murphy, F. R. Gutierrez, R. G. Levitt, and W. M. Thompson, "Oblique views of the heart by magnetic resonance imaging," *Radiology*, vol. 155, pp. 81–85, Apr. 1985.

[22] B. Suits and D. Wilken, "Improving magnetic field gradient coils for nmr imaging," *Journal of Physics E: Scientific Instruments*, vol. 22, no. 8, pp. 565–571, 1989.

[23] D. J. Huber, E. Mueller, and P. Heubes, "Oblique magnetic resonance imaging of normal structures," *Der Radiologe*, vol. 27, pp. 15–20, Jan. 1987.

[24] R. R. Edelman, D. D. Stark, S. Saini, P. Hatt, and J. T. Ferrucci, "Oblique planes of section in mr imaging," *Radiology*, vol. 158, pp. 37–41, Jan. 1986.

[25] D. Ennis and B. Hargreaves, "Gradient hardware constraints — lecture 12a." https://web.stanford.edu/class/rad229/Notes/Lecture-12/Rad229_2020_Lecture_12A_Gradient_Hardware_Constraints.pdf, 2020. Accessed: 2025-07-20.

[26] E. Atalar and E. R. McVeigh, "Minimization of dead-periods in mri pulse sequences for imaging oblique planes," *Magnetic Resonance in Medicine*, vol. 32, pp. 773–777, Dec. 1994.

[27] M. A. Bernstein and P. E. Licato, "Angle-dependent utilization of gradient hardware for oblique mr imaging," *Journal of Magnetic Resonance Imaging*, vol. 4, pp. 105–108, Jan. 1994.

[28] Pulseq Developers, "Pulseq: Open-source MRI pulse sequence framework." https://pulseq.github.io/, 2025. Accessed: July 20, 2025.

[29] PulSeq Developers, "Pulseq file format specification." https://pulseq.github.io/specification.pdf. Accessed: 2025-07-20.

[30] K. Takeda, "A highly integrated fpga-based nuclear magnetic resonance spectrometer," *Review of Scientific Instruments*, vol. 78, no. 3, p. 033103, 2007.

[31] G. N. Holland and J. R. MacFall, "An overview of digital spectrometers for mr imaging," *Journal of Magnetic Resonance Imaging*, vol. 2, no. 2, pp. 241–246, 1992. Instrumentation Subcommittee of the SMRI Basic Science Council.

[32] J. Yun, J. Yu, T. Hongyan, and L. Gengying, "A complete digital radio-frequency source for nuclear magnetic resonance spectroscopy," *Review of Scientific Instruments*, vol. 73, pp. 3329–3331, 09 2002.

[33] C. Job, J. Zajicek, and M. F. Brown, "Fast field-cycling nuclear magnetic resonance spectrometer," *Review of Scientific Instruments*, vol. 67, pp. 2113–2122, 06 1996.

[34] L. Gengying and X. Haibin, "Digital quadrature detection in nuclear magnetic resonance spectroscopy," *Review of Scientific Instruments*, vol. 70, pp. 1511–1513, 02 1999.

[35] D. J. Adduci and B. C. Gerstein, "Versatile pulse programmer for nuclear magnetic resonance," *Review of Scientific Instruments*, vol. 50, p. 1403, Nov. 1979.

[36] J. Dart, D. P. Burum, and W. K. Rhim, "Highly flexible pulse programmer for nmr applications," *Review of Scientific Instruments*, vol. 51, pp. 224–228, 02 1980.

[37] P. P. Stang, S. M. Conolly, J. M. Santos, J. M. Pauly, and G. C. Scott, "Medusa: A scalable mr console using usb," *IEEE Transactions on Medical Imaging*, vol. 31, no. 2, pp. 370–379, 2012.

[38] S. M. Anand, "Ocra: A low-cost, open-source fpga-based mri console capable of real-time control," m.eng. thesis, Massachusetts Institute of Technology, 2018. Department of Electrical Engineering and Computer Science.

[39] V. Negnevitsky, F. Pozo, Y. Kulikova, P. Masson, B. D. de Vos, K. P. Pruessmann, and D. O. Brunner, "Marcos, an open-source electronic control system for low-field mri," *Journal of Magnetic Resonance*, vol. 350, p. 107424, 2023.

[40] H. Wang, F. Lou, Y. Huang, Y. Gao, and X. Zhang, "Extending the marcos: A 4-rx open-source mri console for low-, mid-, and high-field systems," *IEEE Journal of Electromagnetics, RF and Microwaves in Medicine and Biology*, pp. 1–9, 2025.

[41] K. Lin, M. Wang, Y. Wang, H. Li, and W. Wang, "A multi-channel mri console for ultra-high field up to 14 t," *Review of Scientific Instruments*, vol. 96, p. 034702, 03 2025.

[42] D. Gisselquist, "Breaking all the rules to create an arbitrary clock signal." `https://zipcpu.com/blog/2019/06/28/genclk.html`, June 2019. Accessed: 2025-07-20.

[43] AMD Inc., "Kcu105 evaluation kit." `https://www.amd.com/en/products/adaptive-socs-and-fpgas/evaluation-boards/kcu105.html`, 2025. Accessed: 2025-07-20.

[44] AMD Inc., "Oserdese3 primitive." https://docs.amd.com/r/2021.1-English/ug974-vivado-ultrascale-libraries/OSERDESE3, 2021. Accessed: 2025-07-20.

[45] S. Tao, P. T. Weavers, J. D. Trzasko, Y. Shu, I. Huston, John, S. K. Lee, L. M. Frigo, and M. A. Bernstein, "Gradient pre-emphasis to counteract first-order concomitant fields on asymmetric mri gradient systems," *Magnetic Resonance in Medicine*, vol. 77, pp. 2250–2262, June 2017.

[46] E. Katznelson and P. R. Harvey, "Modular whole-body gradient coil comprising first and second gradient coils having linear gradients in the same direction." US Patent No. 5736858, 1998.

[47] C. Juchem, T. W. Nixon, S. McIntyre, D. L. Rothman, and R. A. de Graaf, "Magnetic field modeling with a set of individual localized coils," *Journal of Magnetic Resonance*, vol. 204, no. 2, pp. 281–289, 2010.

[48] M. Takrimi and E. Atalar, "A z-gradient array coil with a dedicated active-shielded array coil for mri," *Magnetic Resonance in Medicine*, vol. 88, no. 6, pp. 2718–2731, 2022.

[49] M. Takrimi and E. Atalar, "Minimization of eddy power loss in the cryostat for a z-gradient array coil driven by an arbitrary pulse sequence: An electromagnetic approach," *Magnetic Resonance in Medicine*, vol. 91, pp. 1225–1238, nov 2023.

[50] R. Babaloo and E. Atalar, "Minimizing electric fields and increasing peripheral nerve stimulation thresholds using a body gradient array coil," *Magnetic Resonance in Medicine*, vol. 92, pp. 1290–1305, apr 2024.

[51] AMD Inc., "Carry8 primitive." https://docs.amd.com/r/2021.2-English/ug974-vivado-ultrascale-libraries/CARRY8, 2021. Accessed: 2025-07-20.

[52] L. Wang, Y. Tong, X. Qin, W.-Z. Zhang, X. Rong, and J. Du, "A field-programmable-gate-array based high time resolution arbitrary timing generator with a time folding method utilizing multiple carry-chains," *Review of Scientific Instruments*, vol. 92, p. 014701, 01 2021.

[53] M. E. Öztürk, R. Babaloo, and E. Atalar, "Pulse sequence design for gradient arrays in mri," in *ISMRM Annual Meeting*, 2024.

[54] M. Loecher, "Gropt: Real-time convex optimization of gradient waveforms for mri." `https://github.com/mloecher/gropt`. Accessed: 2025-07-20.

Appendix A

A.1 RF Submodules of the RF Controller Subsystem

To save space and avoid repetitions, some of the submodule block diagrams of the RF controller from Chapter 3.3 are added to this appendix section.

A.1.1 RF Chain Submodule

In Fig. A.1, the internal structure of the *RF Chain* submodule is presented. *RF Chain* submodule manages the configuration inputs of the RF power amplifiers developed by Ege Aydın. The `PLLAutoMMCM` IP block takes in the output of the fractional clock and filters it using the `MMCM` primitive. For better performance in the class-E amplifiers, the outputs of the `MMCM` are configured as an LVDS pair with 70% duty cycle on the high side and 30% duty cycle on the low side, with half the frequency. Then, by gating according to the **unblank** signal, the filtered and altered clock signals are mapped to the external RF power amplifiers. However, since the input-output map is highly specific to our group, we deemed it redundant to place it inside the full text.

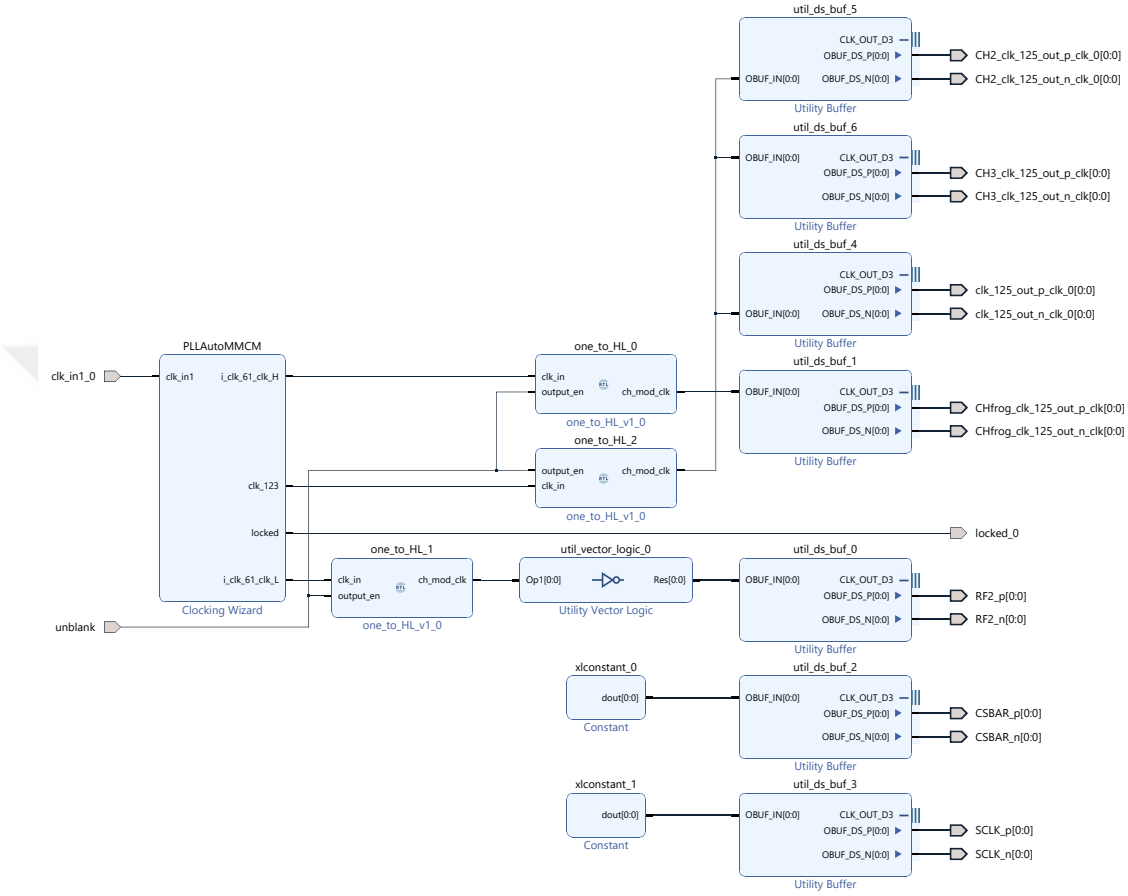


Figure A.1: Internal structure of the RF Chain submodule

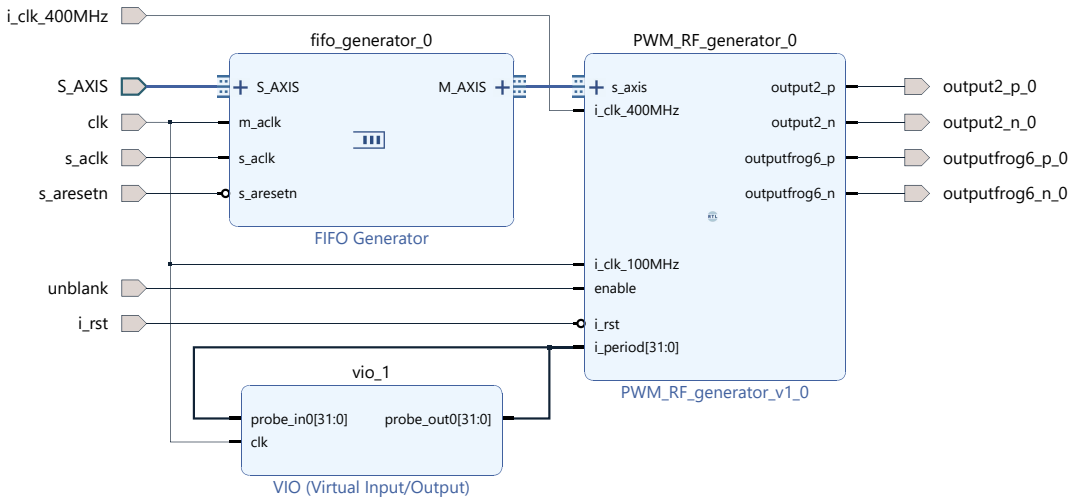


Figure A.2: Internal structure of the Envelope PWM submodule for the RF Controller subsystem

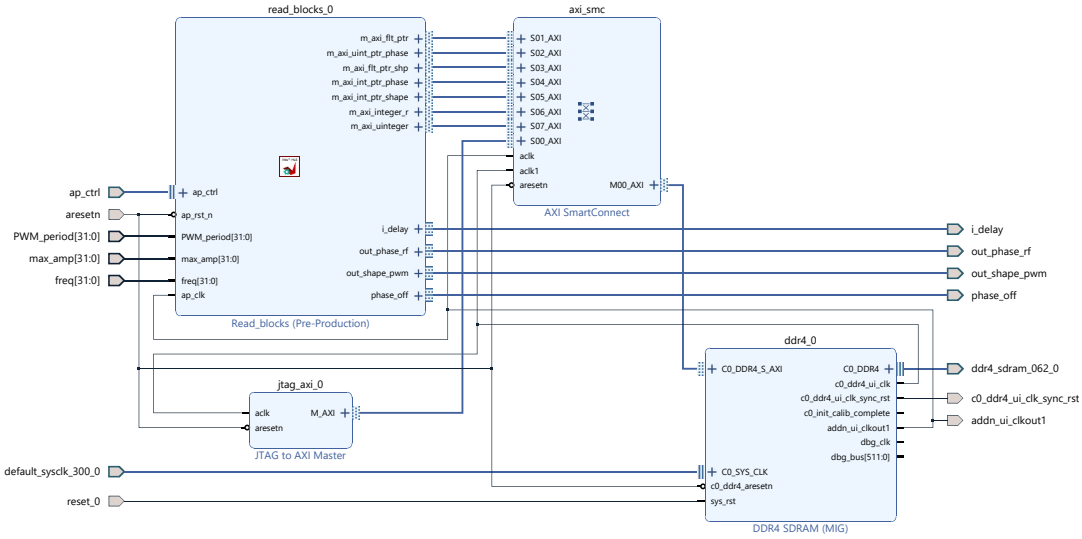


Figure A.3: Internal structure of the Memory Interface submodule for the RF Controller subsystem

A.1.2 Envelope PWM Submodule

In Fig. A.2, the internal structure of the *Envelope PWM* submodule is displayed. The FIFO buffer manages the clock domain crossing between the 300 MHz memory clock and the 10 MHz MR clock. Then the signal is streamed onto the `PWM RF Generator` IP block, where the PWM outputs are generated at 800 MHz output clock with the help of the `OSERDESE3` primitive. The synchronization with the RF clock synthesis, and the external MRI is handled through hard-coded delay blocks and the **unblank** signal.

A.1.3 Memory Interface Submodule

As for the final submodule, *Memory Interface* is shown in Fig. A.3. *Memory Interface* in the RF controller operates similarly to the same submodule inside the gradient controller, as shown in Fig. 3.6. The memory is first programmed through the JTAG interface. Then the `Read blocks` IP block reads, processes, and transmits the time-resolved information downstream to the rest of the system.

A.2 Legacy Feedforward Model of the Array Mapper Submodule

The *Array Mapper* submodule in this work inherits a key functionality from the previous work of Aydın [14]. As described in Chapter 3.2.3, the projection of conventional sequences onto the array channels is not as straightforward. Depending on the amplifier system, the calculated amplitude values may require additional adjustments to compensate for the time-varying negative effects of channel impedances, such as resistance, inductance, or mutual inductance.

In theory, it is possible to convert the gradient array coil system to a feedforward model, including the amplifiers, wires, and multi-channel coils. Such a model can predict the time-resolved physical behavior of the generated fields in response to the input amplitudes. As a result, distortion-free currents can be generated through purely feedforward control, eliminating the need for costly feedback circuitry within the gradient amplifiers.

In practice, realizing such a model is very difficult, but not impossible. Field probes, temperature sensors, and other system monitoring devices can be employed to construct a comprehensive feedforward model. These sensor values can then be used to compensate for distortions in real time. To support such future developments, the current *Array Mapper* implementation includes a placeholder for the integration of a feedforward model.

Figure A.4 illustrates how we preserved the legacy feedforward model from [14]. In this simple demonstration, the impedance of each channel is set to $R = 1\,\Omega$ and $L = 150\,\mu H$, assuming no mutual inductance. As for the current weight matrix, the previous double Maxwell configuration for z-coils is used,

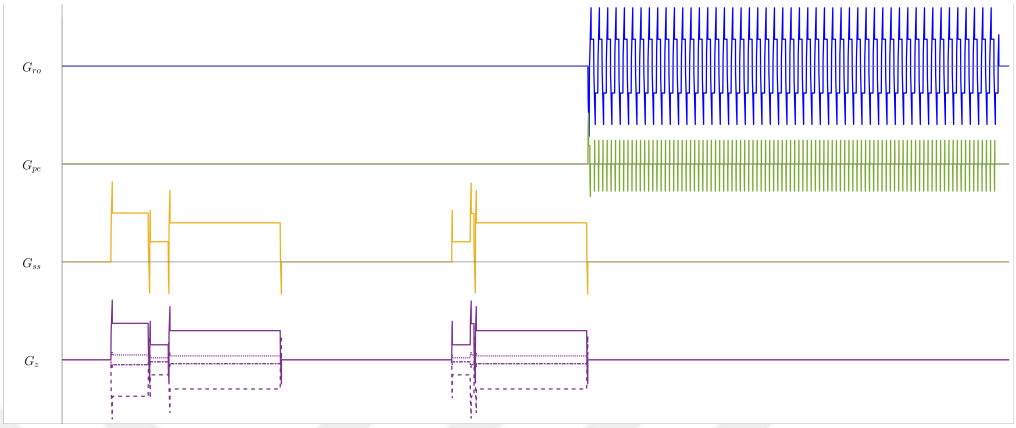


Figure A.4: Simulation results of the *Array Mapper* submodule. The array currents are projected from the G_{ss} using the current weight matrix of [0.746, 0.1, -0.1, -0.746] to the corresponding voltage waveforms with channel parameters of: $R = 1\ \Omega$, $L = 150\ \mu H$.

$$P = \begin{bmatrix} 0 & 0 & 0.746 \\ 0 & 0 & 0.1 \\ 0 & 0 & -0.1 \\ 0 & 0 & -0.746 \end{bmatrix}$$

After feeding the amplitude and slope values of a conventional pulse sequence, the *Array Mapper* submodule calculated the corresponding voltage values for individual array channels and obtained the waveforms. Since the feedforward model is programmed into the FPGA memory before the experiment, on-the-fly adjustments are very much plausible.

Appendix B

B.1 An Alternative Solution to the RF Synthesis

In this appendix chapter, we propose a design strategy to resolve the system imperfections of the RF controller subsystem.

The accumulator in the fractional clock divider holds a 32-bit increment value, given by:

$$g_m[n] = 2^b \Delta t(f_0 n + \Delta f_m n'_m) + \frac{2^b}{2\pi}(\phi_m[n] + \phi_{0_m})$$

In the current system, only the MSB of the increment value $g[n]$ is used to generate the clock signal. Therefore, when the fractional clock divider generates a clock signal, the duty cycle is not fixed and varies across different clock periods. This causes a phase noise, which in turn generates unwanted jitter in the output. However, the remaining 31 bits hold additional information regarding the phase of the current time step. It is, in theory, possible to extract this phase information and correct the jitter that occurs due to the limited output clock frequency of the system.

As a proof of concept, a simulation that utilizes 9 bits of the accumulator was designed. The phase information from the additional bits was extracted. Since

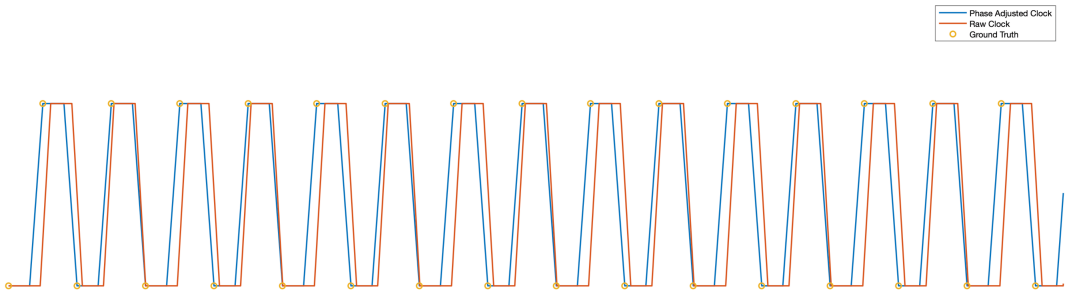


Figure B.1: Comparison of raw clock output and phase-adjusted clock output. The circles in the corner depict the exact time at which the RF clock must be generated to be at the Larmor frequency.

the phase noise is prominent in the clock edges, where the signal switches from high to low or low to high, the transitioning time step was removed. Instead, an output delay, calculated from the phase of the transition, is added to the next time step. Thus, instead of overshooting the phase value and corrupting the duty cycle, a time step is sacrificed, and the next time step is precisely adjusted through fine-grained delay elements.

First, the phase information from the additional bits must be investigated to calculate the exact delay value. Each extra bit in the increment value of $g[n]$ divides the time step of Δt by multiples of 2. For example, the most significant bit of $g[n]$ corresponds to a time step of Δt, whereas the 11^{th} MSB of $g[n]$ corresponds to a time step of $\Delta t \cdot 2^{-10}$. Therefore, the 11-bit accumulator value represents the phase information at the picosecond-level ($1.25\,ns \cdot 2^{-10} = 1.22\ ps$).

The simulation extracts the 9-bit information from the accumulator (delay step resolution of 4.88 ps), skips the clock edge time step, and instead holds the previous value for the calculated delay time before transitioning to the next state of either high or low. Utilizing this fine-grained delay chain, the phase noise is significantly reduced, and duty cycles are almost fixed. The resulting figure can be seen in Fig. B.1.

It can be seen in Fig. B.2 that, unlike the phase-adjusted clock, the high-frequency components of the raw clock output are still present at 62 and 183 MHz. This also shows the reduction of the phase noise and duty cycle adjustment

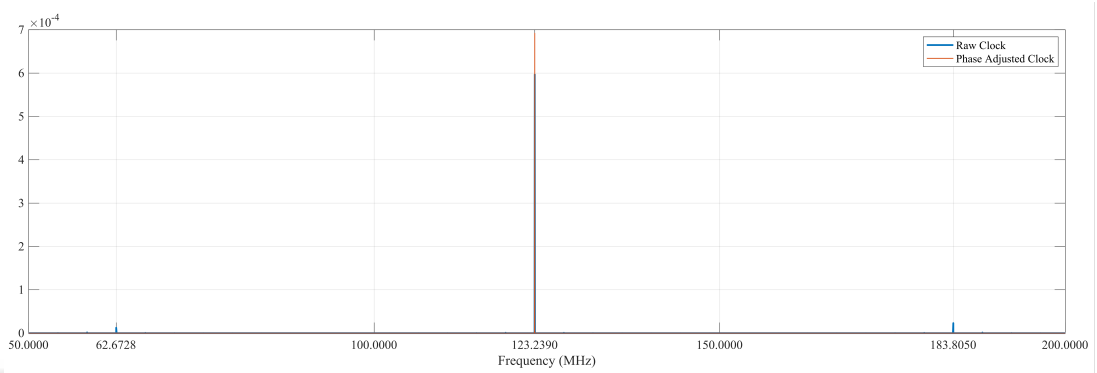


Figure B.2: Frequency response of both the raw and phase-adjusted clock outputs

that the fine-grained delay chain provides.

While the simulations validate the concept, the hardware implementation of the proposed methodology is a tough challenge. It requires dedicated tap delay lines that can cover a span of 1.25 ns with 5 ps step sizes, which, if not careful, can consume a large amount of FPGA resources. However, it is also possible to use the time folding method, proposed by Wang et al. [52] in their arbitrary time generator circuitry.

Their work utilizes two `CARRY8` primitives whose inputs are loaded simultaneously using FIFO buffers. A clock schedules these inputs, allowing continuous buffering of clock transitions. By concatenating multiple such lines, they cleverly increase the effective time resolution of the delay line down to 45 ps.

In our case, a similar approach can be realized in the future. The output of the fractional clock divider, which was initially connected to the `OSERDESE3` primitive, can directly be fed to an `MMCM`. This way, both the adequate time resolution can be increased dramatically, and the interferences caused by the physical loop-back wiring can be eliminated. However, this topology demands meticulous design, as each routed path contributes to the overall delay. It is quite a challenge to manually route such paths under strict time constraints. All being said, the effort is worth it for the benefits. The resulting RF controller can effectively generate its own arbitrary RF clock without any nonlinear components such as DAC or mixers. Each design step can be digitally controlled. Moreover, the design can

be scaled up with minimal budget cost.