Journal of Magnetic Resonance 299 (2019) 74–92

Contents lists available at ScienceDirect

Journal of Magnetic Resonance

journal homepage: www.elsevier.com/locate/jmr

An autonomous, highly portable NMR spectrometer based on a low-cost System-on-Chip (SoC)

David Ariando *, Cheng Chen, Mason Greer, Soumyajit Mandal

Case Western Reserve University, 10900 Euclid Avenue, Cleveland, OH 44106, USA

ARTICLE INFO

Article history:
Received 17 September 2018
Revised 5 December 2018
Accepted 7 December 2018
Available online 15 December 2018

Keywords:
NMR spectrometer
System-on-Chip (SoC)
Auto-tuning
Autonomous operation

ABSTRACT

This paper describes the development of a portable and self-optimizing NMR spectrometer based on a miniaturized custom analog front-end and a System-on-Chip (SoC)-based digital back-end. The SoC integrates a field-programmable gate array (FPGA) fabric with a hard processor running a Linux operating system, thus enabling fully-autonomous operation without the need for an external computer. In the proposed approach, data captured by the FPGA fabric during regular operation is transported to the hard processor using an integrated on-chip bus for further processing. The processed results are then used to automatically estimate parameter values that optimize a suitable cost function, such as signal-to-noise ratio (SNR) per unit time. Finally, the optimized values of both electrical and NMR-related tuning parameters (e.g., preamplifier gain and frequency response, pulse length and amplitude, operating frequency, etc.) are programmed back into the front-end and back-end hardware. Experimental NMR results from various samples in a ~0.1 T permanent magnet are presented to verify the operation of the proposed spectrometer. These demonstrate on-board Laplace inversion and automated frequency tuning to compensate for temperature changes. Preliminary ^{14}N NQR results are also presented.

1. Introduction

Nuclear magnetic resonance (NMR) has become a common analytical method because of its broad spectrum of applications, which include chemistry, biomedical imaging, food science, and down-hole measurements of hydrocarbon reservoirs. In some of these applications, a powerful stand-alone computer is available for post-processing and visualization of experimental data acquired by the NMR spectrometer. However, in other cases, such computers may not be available due to a limited size and/or power budget. These include both well-established "field" applications, such as down-hole measurements [1], and also emerging ones where the device should be small enough to be carried anywhere, such as soil analysis [2,3] and point-of-care (PoC) diagnostics [4].

Recent implementations of custom-designed low-cost NMR spectrometers for such applications have been primarily based on field-programmable gate arrays (FPGAs) [5]. FPGAs are used in a variety of modern scientific instruments due to their widespread availability, inherently parallel architecture, and ability to handle precision timing requirements. Timing-critical parts of such instruments, e.g., the NMR pulse programmer, are therefore usually constructed within an FPGA fabric. On the other hand, non-critical parts, e.g., blocks that compute experimental parameters and post-process the results, can be implemented using a variety of devices, such as digital signal processors (DSPs) [6,7], or personal computers (PCs) [8–14]. However, other implementation approaches are also possible. For example, the timing-critical parts of NMR specrometers can be implemented using a microcontroller (MCU) instead of an FPGA [15,16]. MCUs have certain advantages over FPGAs for such applications: they are generally less expensive, consume less power, and can be programmed using high-level languages (such as C) instead of hardware description languages (such as Verilog or VHDL), which results in rapid development times. However, they have inherently serial architectures, which results in issues with maintaining precise and deterministic timing during complex operations (e.g., due to interrupt latency). Yet another approach is based on developing custom application-specific integrated circuit (ASICs) to implement the required hardware [17]. This approach allows complete NMR spectrometers to be miniaturized down to the chip level, which is advantageous in size- and power-constrained applications such as PoC diagnostics and down-hole well-logging. However, ASICs have long and expensive development cycles due high fabrication costs and the need to use specialized design tools. Thus, they are not suitable for many low-cost applications.

* Corresponding author.
E-mail addresses: dja62@case.edu (D. Ariando), cxc717@case.edu (C. Chen), mkg53@case.edu (M. Greer), sxm833@case.edu (S. Mandal).

https://doi.org/10.1016/j.jmr.2018.12.007
1090-7807/

NMR signals acquired in strong and uniform magnetic fields have narrow bandwidths and are sensitive to changes in (i) external environmental variables such as temperature, (ii) sample properties such as conductivity, and (iii) spectrometer operating parameters such as frequency. As a result, the optimum signal acquisition conditions tend to change over time. Thus, it becomes necessary to carry out a tuning procedure before acquisition in order to obtain optimum parameter values. Tuning is also important in order to find the pulse sequence parameters that are optimal for analyzing an unknown sample. Finding these software parameters can take a long time, since it generally requires sweeps over the frequency range, pulse length, pulse amplitude, etc. This work aims to automate the process of tuning the NMR measurement parameters before acquisition.

Many, if not all, of the NMR spectrometers cited above rely on a PC as the system controller; a program on this computer typically allows the user to change the pulse program parameters and view experimental results. The controller can thus be used to tune both hardware and software parameters. Given a particular permanent magnet, computing the Larmor frequency of the sample is typically the first step during tuning. In order to obtain quantitative NMR results, hardware tuning then needs to be performed for every frequency-dependent component in the spectrometer, e.g. the impedance-matching network, the transmit path (including the power amplifier), and the receive path (including low-noise preamplifier). This process can be fairly tedious in practice, since multiple parameters need to be individually optimized. Moreover, it may require sophisticated measurement equipment, e.g., oscilloscope, network analyzer, etc. Finally, the software parameters are tuned, e.g., by monitoring the signal amplitude versus pulse length or amplitude. The entire process becomes even more complicated if the spectrometer needs to be used with multiple sample coils (e.g., with different geometries) and/or multiple nuclei. In this case, even though software tuning can be automated relatively easily by writing a specialized program, hardware tuning becomes extremely slow and generally requires a tedious amount of human intervention. For portable applications, the task becomes even more difficult due to the absence of measurement equipment, which ultimately limits the usage of NMR spectrometers in various field applications.

The recent emergence of systems-on-chip (SoCs) equipped with both hard processors and FPGAs can provide a solution to these problems. Such hybrid devices provide high-level programming capabilities that are similar to that of a PC, while also preserving the precision timing capabilities of an FPGA. The hard processor is usually an embedded ARM core; such cores are in widespread use within mobile devices and embedded systems (e.g., single board computers such as the Raspberry Pi). Chip manufacturers can readily license these processors from the designer (ARM Holdings) as so-called intellectual property (IP) cores and then integrate them together with memory, programmable logic, and peripherals into a wide variety of powerful SoC and system-on-module (SOM) designs [18,19]. This paper describes how a commercial low-cost SoC can be used to realize a miniaturized yet powerful and highly-programmable NMR spectrometer. The chosen chip integrates a dual-core ARM processor that is capable of running an embedded Linux operating system (OS) together with (i) programmable logic (a Cyclone-V FPGA fabric), (ii) a limited amount of memory, and (iii) on-chip buses for data transfer between the ARM processor and the FPGA fabric. The ability to run an embedded OS enables us to use high-level programming languages (C, Python, shell scripts, etc.) directly inside the spectrometer. Other advantages of the OS include its ability to handle a large amount of storage, use different networking protocols (including wireless ones), and run a graphical user interface. The OS even makes it possible to switch between different FPGA implementations on-the-fly, which may be very useful in future work.

We also take advantage of the SoC's capabilities to make the NMR spectrometer portable and self-contained. In particular, the device incorporates abilities to adapt itself to different conditions by performing self-tuning to find its optimum configuration for a given environment, sensor, and sample. In this paper, we start by discussing potential new applications for autonomous NMR spectrometers in Section 2. We then discuss the hardware architecture and demonstrate the basic capabilities of the spectrometer by performing basic NMR measurements in Section 3, Section 4, and Section 5. Various configurable aspects of the spectrometer are then tested and reported. In Section 6, we present a more complicated scenario in which the spectrometer autonomously adapts its hardware configuration to obtain the maximum possible SNR from the sample as the ambient temperature changes with time over a wide range. Finally, we describe and implement a dynamic optimization method for NMR signal acquisition which exploits feedback from the spectroscopic measurements, leading to a self-optimized experiment that automatically adapts to different test samples. Section 7 concludes the paper and discusses directions for future work.

2. Applications of autonomous NMR systems

Autonomous NMR spectrometers have advantages in applications such as in-line process monitoring, soil sensing, and well logging tools. In-line process monitoring applies across many industries and is used to continuously monitor products during the manufacturing process to ensure quality control. Near infrared and Raman spectroscopy have been used for in-process monitoring, but the use of NMR is less common [20,21]. Challenges for in-line NMR to be applied as an industrial sensor include implementation cost, time required for sample polarization, and the fact that individual samples require different pulse parameters to maximize measurement quality (e.g., SNR per unit time). However, because of its nondestructive nature and sensitivity to changes in molecular structure, NMR has the potential to be a powerful tool in such applications. Our autonomous NMR system can operate as an "edge device" in an Internet of Things (IoT) network, and is thus attractive for industrial use. An illustration of how our system may be used as an in-line process monitoring system in the food industry can be seen in Fig. 1. In this scenario, a food product will constantly flow through the NMR device during production or packaging. The device will autonomously maximize measurement

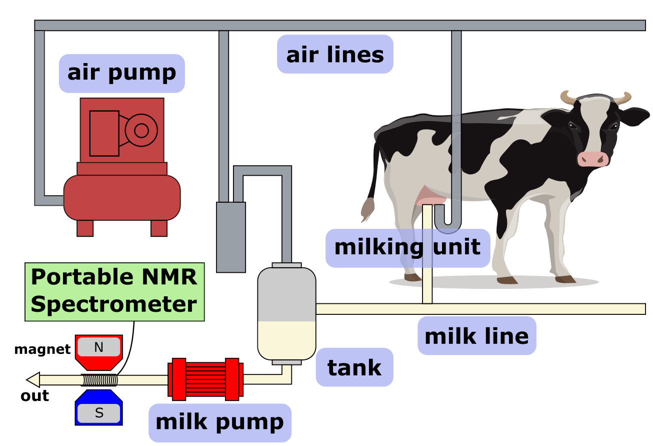

Fig. 1. An example of in-line NMR process monitoring in the food industry. Here the spectrometer monitors the quality of milk in a production line.

quality to continuously monitor signal features (e.g., initial amplitude or T_2) that provide information on product quality.

Soil quality is very important to the agricultural industry, animal life, and Earth science. Thus, there exists a need for accurate and continuous monitoring of soil. Three key soil quality metrics are water, carbon, and phosphorus content, all of which can be detected through NMR [22]. Moisture content measurements can help farmers optimize water given to plants [23]. Another important soil quality metric is total carbon content of the soil, which is useful for agriculture and monitoring CO_2 emissions from soil that contribute climate change [24]. Additionally, phosphorus levels are important to surrounding bodies of water, as rapid phosphorus loss from soil can contribute to eutrophication [25]. NMR tools have not been commonly used for continuous soil monitoring due to limited spatial resolution, high cost, large size, and complexity of data collection. The proposed miniaturized autonomous spectrometers can address this issue by enabling the deployment of large-scale NMR sensor networks that wirelessly transmit the acquired data to a hub for aggregation and further processing. Conventional computers are not suitable for implementing such NMR-based sensor nodes due to high cost and power consumption, whereas lower-cost alternatives such as ultra-low-power microcontrollers have severe memory limitations. SoCs provide a good compromise between these extremes: A confirms that the proposed SoC-based implementation has about 10$\times$ lower power consumption than typical computers.

In the examples above, data transfer bandwidth to the end user is not a large issue as Wi-Fi, Bluetooth, Ethernet, or cellular cervices can be used for high speed data transfer. However in some applications, such as well logging, data bandwidth becomes very limited. For example, the most commonly used data transfer scheme used in well logging is mud pulse telemetry, which can transmit data at a speed of only $\sim$20 bits per second (bps) [26,27]. In such scenarios, it is beneficial to have all data processing done on the device itself and transmit the minimum amount of data back to the surface. The on-board ARM processor in our system allows (i) data collection parameters to be autonomously reconfigured to optimize performance metrics (e.g., SNR per unit time); (ii) data inversions to be performed to obtain T_1, T_2, or D distributions if needed; and (iii) other data processing steps to be run directly on the device itself, thus eliminating the need to transfer raw data in real-time.

Data security is very important when performing any type of chemical measurement as the composition of the material under question may be proprietary or confidential. When transferring raw data to a host computer or the cloud, there exists an inherent risk of data interception. Data encryption can provide some sort of security but it is not fool proof [28]. By doing all data processing locally on the device itself, the data can be abstracted away from its raw form before transfer to a server, thus making the system much more robust to data breaches. For example, in some scenarios the user only needs to know whether the sample is acceptable or not. In this case, raw data can be processed and kept on the local device, making it very resilient to data attacks, and only the final binary decision (i.e., category) can be transmitted to the user.

3. Analog front-end design

Fig. 2 shows a high-level block diagram of the proposed NMR spectrometer. The first major component is an analog front-end (AFE) board that contains a transmitter (low-pass filter and power amplifier), a receiver (low-noise preamplifier and second-stage amplifier), an analog-to-digital converter (ADC), and a passive duplexer. The second component is a digital back-end board (Terasic DE1-SoC Altera development board) that contains (i) the NMR pulse program, ADC driver, and first-in first-out (FIFO) buffer on the FPGA side, and (ii) Linux operating system on the ARM processor side. The third component is a digitally-tunable impedance matching network. The final component is a variable-gap permanent magnet, the sample coil probe within a 3-D printed plastic holder, and a temperature sensor attached to the magnet. In this section, we describe the tunability of the proposed AFE and matching network. More circuit details are provided in Appendix B.

Our transmitter design uses a custom class-A push-pull linear power amplifier with differential inputs and single-ended output. The design allows the user to select between two output stages with different power handling capabilities (2.5 W up to 40 MHz, or 10.6 W up to 11.6 MHz). The input is directly driven by two FPGA pins that generate differential square-wave signals, which allows us to avoid the use of high-speed digital-to-analog converter (DAC). Harmonics of the square-wave are filtered using a multiple-feedback differential low-pass filter. This approach requires the low-pass filter to be programmable in order to operate over broad frequency ranges, as required in several applications of the autonomous NMR system. The use of switched resistors or digitally-controlled potentiometers for implementing such tuning is one of many possible improvements to the current system that can studied in future work. Further details about the transmitter implementation are discussed in Appendix B.1.

The duplexer used in this work is based on a well-known passive lumped duplexer design that was originally proposed by McLachlan [29]. The circuit has low quality factor (Q) which provides sufficient bandwidth and eliminates the need for digital tuning. Further details about the duplexer implementation are discussed in Appendix B.2.

A series-shunt capacitive impedance matching network is used in this work, with a target operating frequency range of 3.5–4.5 MHz. The typical Q value of the coil at the targeted frequency results in a bandwidth of only $\sim$100 kHz. To cover the whole frequency range, a switched-capacitor matching network is used, based on the work of Chen et al. [30]. The circuit is discussed in detail in Appendix B.3. The matching network uses relays to switch capacitors, thus allowing the tuning to be digitally-controlled from software. A directional-coupler-based reflection measurement is also implemented in order to quantify impedance matching of the digitally-tunable matching network, effectively closing a feedback loop for automatic tuning of the matching network. The details of the circuit are discussed in Appendix B.4.

The proposed preamplifier circuit uses a common-gate topology using a low-noise junction field-effect transistor (JFET) with tuned output load. The output load is an LC circuit consisting of a fixed inductor in parallel with varactors (i.e., voltage-controlled capacitors). In this case, both the varactors and preamplifier input bias voltage are digitally controlled using a dual-output digital-to-analog converter (DAC), allowing the input impedance and frequency response to be set using software. Further details of the preamplifier are discussed in Appendix B.5.

In order to digitize the amplified receiver outputs, a 14-bit, high-speed (25 Msps) parallel-output ADC is used. A high-speed *parallel* ADC is preferable over the *serial* ADC we used in our previous work [31] in order to satisfy Nyquist sampling over the expected 3–5 MHz Larmor frequency range, thus eliminating the need for undersampling techniques such as bandpass sampling. Moreover, the parallel output bus allows the clock rate to be significantly lower than for a serial ADC, which simplifies the ADC interface design. Further details about the ADC interface are discussed in Appendix B.6. The ADC clock is digitally programmable using a reconfigurable phase-locked loop (PLL), which allows the sampling frequency to be easily programmed using software.

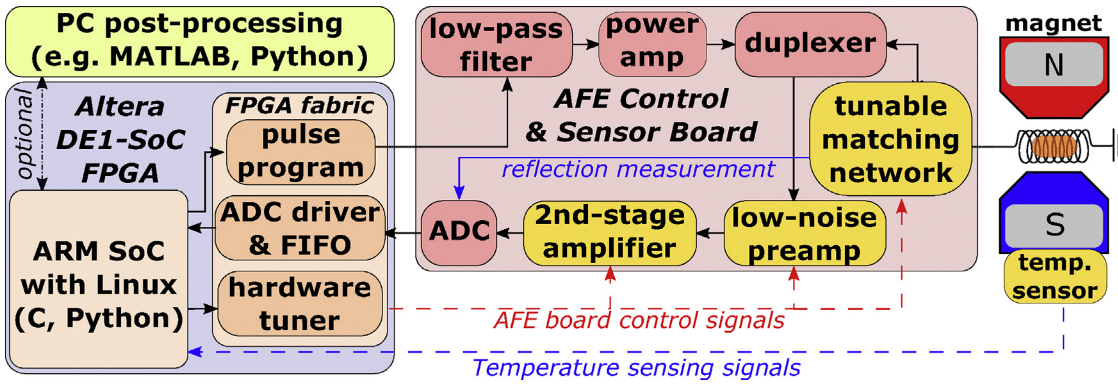

Fig. 2. Simplified block diagram of the proposed autonomous and self-contained NMR spectrometer.

4. Digital back-end design

The digital back-end architecture implemented on the SoC is designed to be highly programmable while also being easy to maintain and update. It is useful to first understand how the data is transported in the system, which is summarized in Fig. 3. The two main segments are implemented on the FPGA fabric and the embedded ARM core, which is also known as the hard processor system (HPS). The FPGA interfaces with the AFE using general-purpose input/output (GPIO) pins that allow it to tune the hardware, run the actual pulse sequence, and temporarily store the acquired data. The FPGA is configured using a hardware description language (HDL) design implemented in Verilog. An Advanced eXtensible Interface (AXI) bus available within the SoC is used for bidirectional communication between the FPGA and the ARM. The ARM processor runs an Ubuntu Linux OS, and the Linux filesystem is used to transfer data between C and Python programs that run low-level NMR primitives and high-level processing, respectively. In addition, a set of HPS input/output (I/O) pins is used to input sensor data, e.g., the magnet temperature recorded by a temperature sensor, directly into the ARM. Finally, Ethernet is used to transfer data to/from an external PC.

We now provide a more detailed description of the data flow. The data samples acquired by the ADC are transported from the AFE to the FPGA via parallel GPIO pins, and then stored inside a streaming FIFO buffer with a maximum length of 128 kB (limited by the amount of on-chip memory available). A direct memory access (DMA) transfer from the FIFO buffer to on-board memory (SDRAM) is also implemented to expand the memory limit to 64 MB. This is enough to store 1.28 s worth of data samples at 25 Msps, which is practically sufficient for most low-field NMR applications. Data inside the FIFO or SDRAM can be accessed by the ARM processor via an AXI/Avalon memory-mapped bus provided by Platform Designer (previously known as Qsys) inside Intel's freely-available Quartus software. In our implementation, a C program running on the ARM is used to transport acquired data from the FIFO to a standard text file in the Linux filesystem. This file can be directly transported to a PC via ethernet if further processing in a more powerful software environment (e.g., MATLAB) is desired, which was our approach during the development phase. However, the availability of Python programming in the Linux OS opens up the possibility of porting most of the high-level NMR functions previously developed for the PC to the embedded ARM processor. These include quadrature down-conversion, matched filtering, noise measurement, reflection measurement, frequency and pulse length sweeps, multi-exponential fitting of echo decay curves, and inverse Laplace transforms (ILTs) for generating $T_1 - T_2$ and D-T_2 maps. In addition, having the ability to analyze the data without transferring it to an external PC, opens up the possibility of performing real-time hardware-software co-tuning especially useful in portable spectrometry applications. In our case, a Python program acts as the main control module and can run NMR experiments by calling executables compiled from various low-level C programs. The latter perform hardware tuning, run various NMR pulse programs, analyze the acquired data generated by the pulse programs, make the decision to change experimental parameters if necessary, iterate the process to get desirable results, and finally either store the results (in a text file) or directly display them to the user (in a graphical format).

The important blocks within the digital back-end are further described in the next few sections.

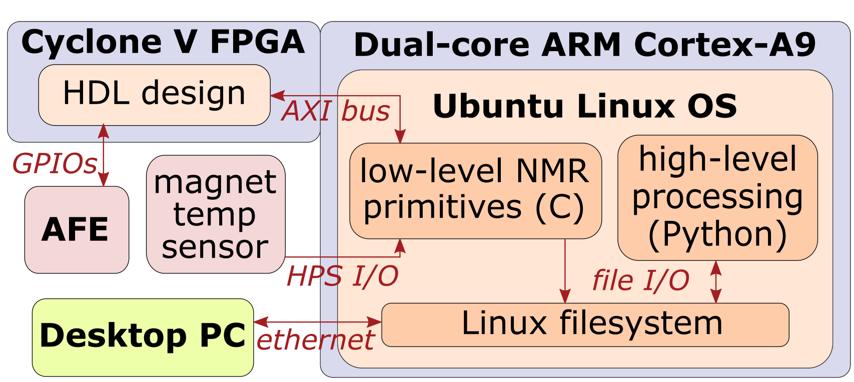

Fig. 3. The overall software flow of the proposed NMR spectrometer. GPIO: General-purpose input/output pin; HDL: Hardware description language.

4.1. Pulse program and ADC controller

The FPGA manages timing-critical operations, which include two fully-synchronous building blocks, namely a pulse program module and an ADC driver with FIFO memory module. The basic pulse program module developed in the FPGA is a generalized version of a Carr-Purcell-Meiboom-Gill (CPMG) pulse sequence [32,33]. The user needs to specify CPMG parameters, e.g. $\pi/2$ pulse length, π pulse length, and echo spacing, in human-readable format, i.e. as input variables with units of μs. These values are automatically converted by a C program into integer values and sent to control registers in the FPGA. More technical details on pulse programming are provided in Appendix C.

All derived integer parameters shown above are sent to the pulse program module in the FPGA via the on-chip AXI bus. This module, which is shown in Fig. 4, uses the output of an on-chip fractional-*N* PLL to generate the operating frequency, i.e., system clock. The PLL provides programmable and precise frequency synthesis with sub-Hz resolution. The pulse program generates two differential output signals with phase selector control to select pulse phases in the rotating frame defined by the system clock, for example the *x*-phase and *y*-phase pulses used in the CPMG sequence shown in Fig. C.27. After this, the overall transmit output signal is enabled or disabled by a control signal also generated by the pulse program. In the receiver path, the data from the ADC is captured by the ADC driver and then stored in a FIFO.

It was experimentally found that allowing the absolute phase of the transmit signal to vary between scans can result in problems. For example, this makes it impossible to directly average data across multiple scans. We can work around this issue by phase-rotating the average echo shape acquired in one scan to match those from previous scans, thus enabling averaging across experiments as described in our previous work [31]. However, this method has a caveat: it is only effective when the amplitude of the NMR signal is larger than that of the unwanted coil ring-down, otherwise the ring-down signal will dominate and the NMR signal will be canceled out after phase cycling. In the current design, this problem is fixed by keeping the absolute phases for all experiments identical, i.e., by synchronizing the system clock, NMR transmit clock (i.e., RF frequency), and the ADC clock. The system clock frequency is chosen to be 16$\times$ and 4$\times$ of the NMR transmit clock and ADC clock, respectively. An ADC clock frequency of 4$\times$ the NMR transmit frequency is chosen to simplify future implementations of quadrature down-conversion within the FPGA, as suggested by Takeda [9]. A system clock frequency of 16$\times$ the transmit frequency is chosen to (i) allow more pulse phases (up to 16) to be generated; and (ii) provide some flexibility during logic design of the pulse program module.

Variations in the absolute phase of the transmitted signal between the refocusing pulse in one scan may also introduce problems, as described by Mandal et al. [34]. In particular, such variations cause additional transverse relaxation, and the effect increases at low Larmor frequencies and high RF power levels. Therefore, it is necessary to synchronize the absolute phase of the transmit signal for every refocusing pulse. In this paper, absolute phase control is performed by only enabling output transmit signals when a counter driven by the system clock counter reaches multiples of 16; this ensures that all pulse lengths and time delays are integer multiples of the NMR transmit clock period (i.e., period of the RF waveform). As a result, pulse lengths will shrink or expand by a small amount during frequency sweeps depending on the closest available match between the lengths specified by the user and the period of the system clock multiplied by 16. Fortunately, such changes in pulse lengths are generally small enough to have no significant effect on the acquired signal. However, it is important to note that we can easily switch to other FPGA designs to avoid this effect if needed; each design is simply stored as a file on the SoC. For example, we can switch to an implementation that decouples the transmit and system clocks, which allows pulse lengths and delays to remain strictly constant as the transmit frequency varies.

4.2. High-level abstraction on the SoC

The second part of the digital back-end is implemented on the ARM processor, which runs a Linux OS. Some low-level software routines (written in C) are used to transform different parameters specified by the user, which are human-understandable, into integer values stored within registers for use by the FPGA program, e.g. pulse length, pulse duty cycle, pulse frequency, echo spacing, experiment spacing, number of echoes per scan, number of samples per echo, phase cycle, and some delay parameters for the acquisition. The C program also provides an interface to transfer the ADC data from the FIFO to the Linux OS via the memory-

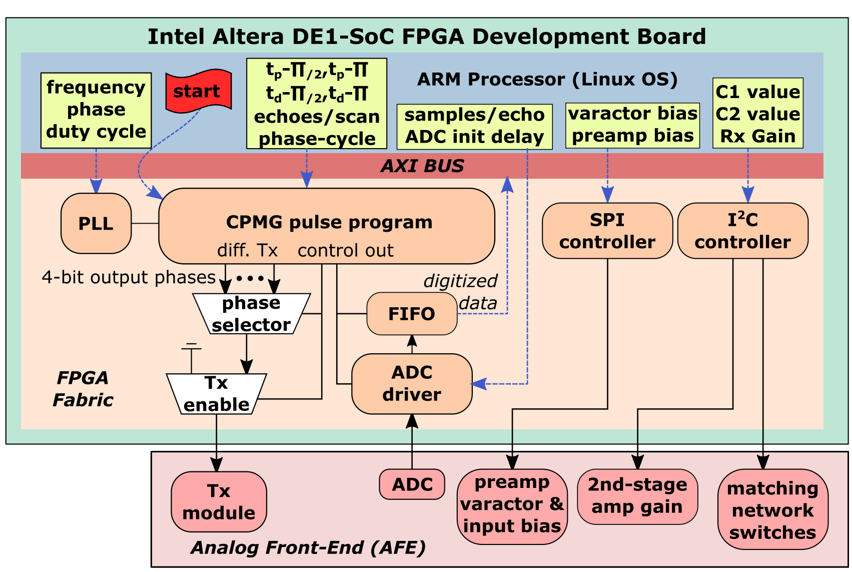

Fig. 4. Block diagram of the pulse program module implemented within the FPGA. The AFE hardware parameters are tuned using two serial buses: an SPI bus and an inter-integrated circuit (I^{2}C) bus.

mapped AXI/Avalon bus. Some higher-level measurements are also defined here, including averaging, frequency sweeps, pulse length sweeps, duty cycle sweeps, inversion recovery experiments, reflection measurements (“wobble” functions), and noise measurements. Python provides a complete set of free and open-source high-level libraries for scientific computing and is thus used for performing further signal processing inside the embedded system, e.g. quadrature down-conversion and matched filtering. Details are provided in Appendix D.

5. Experimental results

5.1. Experimental setup

Fig. 5(a) shows a photograph of the complete NMR spectrometer, which includes DC power supplies (± 15 V, ± 5 V), the Altera DE1-SoC board, the sample coil and coil holder, the permanent magnet, the temperature sensor, and two AFE boards (controller board and sensor board). Note that the system is completely self-contained; the only external equipment used are the two benchtop power supplies, which can be easily replaced with miniaturized modules in future work.

Experiments were carried out using a 3D-printed sample holder that was designed to accommodate standard 5 mm NMR sample tubes. A solenoidal coil (length ≈ 21 mm, diameter ≈ 8 mm, 14 turns) was wound around this holder and glued in place. It was then placed into a miniature variable-gap permanent magnet (EM-8618, Pasco Scientific), as shown in Fig. 5(b). The magnet uses a conventional U-shaped geometry that consists of two cylindrical NdFeB magnet pieces (diameter $= 2.54$ cm) connected by a soft-iron yoke. Flat pole pieces (mild steel, 7.95×4.44 cm) are used to improve field homogeneity within the sample volume. We set the gap to 10.5 mm in order to accommodate the sample holder, which resulted in a field strength of 0.101 T.

5.2. Electrical characterization and tuning capabilities

This section describes the measured characteristics of the AFE. Fig. 6(a) shows the measured output voltage generated across the coil for a pulse length of 5.5 μs and a duty cycle of $D = 0.5$ (i.e., 50%). The matching network was tuned to ensure good matching at the operating frequency, as visible from the shape of the pulse. Fig. 6(b) summarizes the measured peak-to-peak voltage across the coil as a function of the transmitter duty cycle D. The plot also compares the measurement with the theoretical prediction, which is $\propto \sin(\pi D)$; the two are in good agreement.

The preamplifier was characterized using two-port S-parameters measured using a network analyzer. Fig. 7(a) shows the measured reflection coefficient $|S_{11}|$ of the preamplifier as a function of the gate bias voltage V_G (which controls the input transistor's transconductance g_m). The common gate topology of the preamplifier results in relatively broad matching bandwidth for a given V_G, but varying V_G provides additional control. All experiments in this paper use a fixed value of V_G unless otherwise stated; this value was chosen to minimize $|S_{11}|$ around 3.9 MHz (shown as the bold line in Fig. 7(a)). The resulting matching bandwidth (defined as the region for which $|S_{11}| < -10$ dB, corresponding to $< 10\%$ of the incident power being reflected) is around 740 kHz, which is sufficient for most NMR measurements with the chosen magnet.

Fig. 7(b) shows the measured preamplifier gain (estimated from $|S_{11}|$) for a variety of varactor bias voltages while keeping V_G fixed. As expected, the responses are band-pass, thus providing the desired frequency selectivity. Moreover, the center frequency can be tuned over more than an octave (2.3–5.5 MHz), as expected. The maximum gain of 31 dB occurs at 3.9 MHz, which is because the preamplifier is impedance-matched at the input at this frequency. Finally, the peak gain can be tuned to approximately 30 dB over the entire input matching bandwidth of 740 kHz.

Apart from the preamplifier, the reflection coefficient of the probe (coil and matching network), is also of interest. Our current design uses a directional coupler to estimate the probe's reflection coefficient versus frequency. A C program first generates a transmit frequency sweep for this purpose. The reflected signal from the probe (as coupled by the directional coupler) is captured and digitized by the on-board ADC at each frequency step once steady-state conditions have been reached. A fast Fourier transform (FFT) of the digitized signal is then used to estimate the amplitude of the reflected signal at the input frequency, thus resulting in one point per frequency step. Fig. 8 shows the measured results for several combinations of C_1 and C_2. The plot shows that excellent

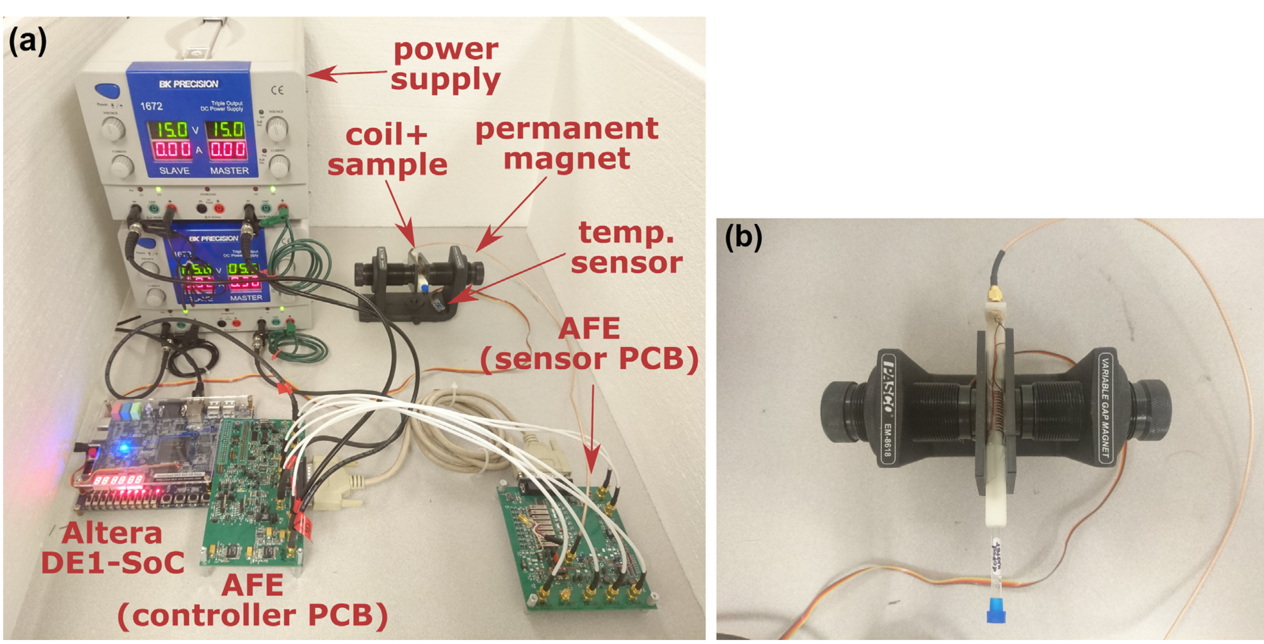

Fig. 5. Photograph of (a) the complete NMR spectrometer, and (b) the magnet and sample holder.

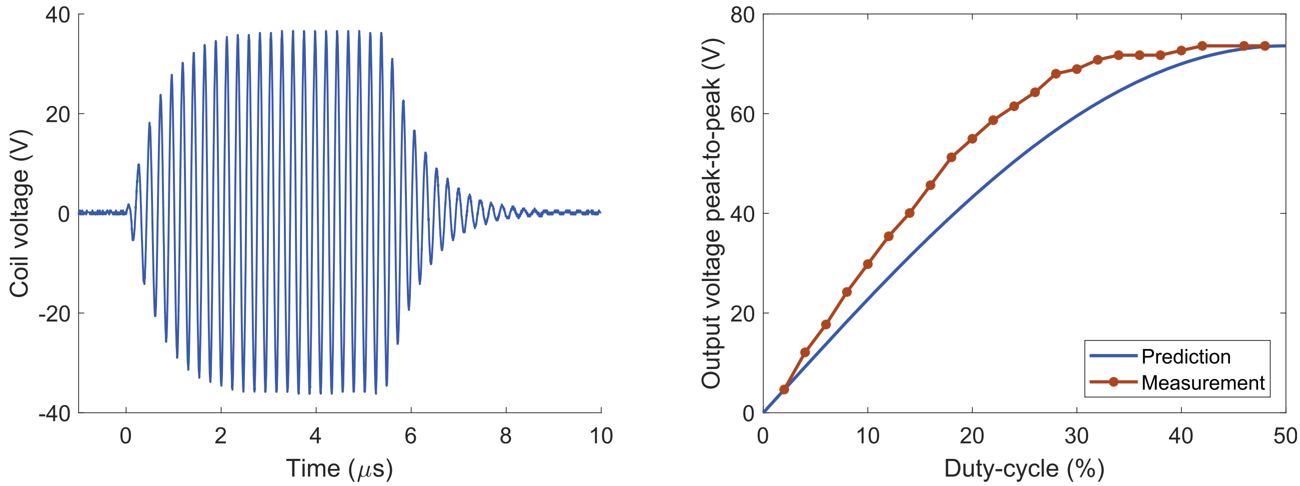

Fig. 6. (a) Measured coil output voltage waveform during a pulse with a transmitter duty cycle of 50%; (b) predicted and measured peak-to-peak coil voltage as a function of the transmitter duty cycle.

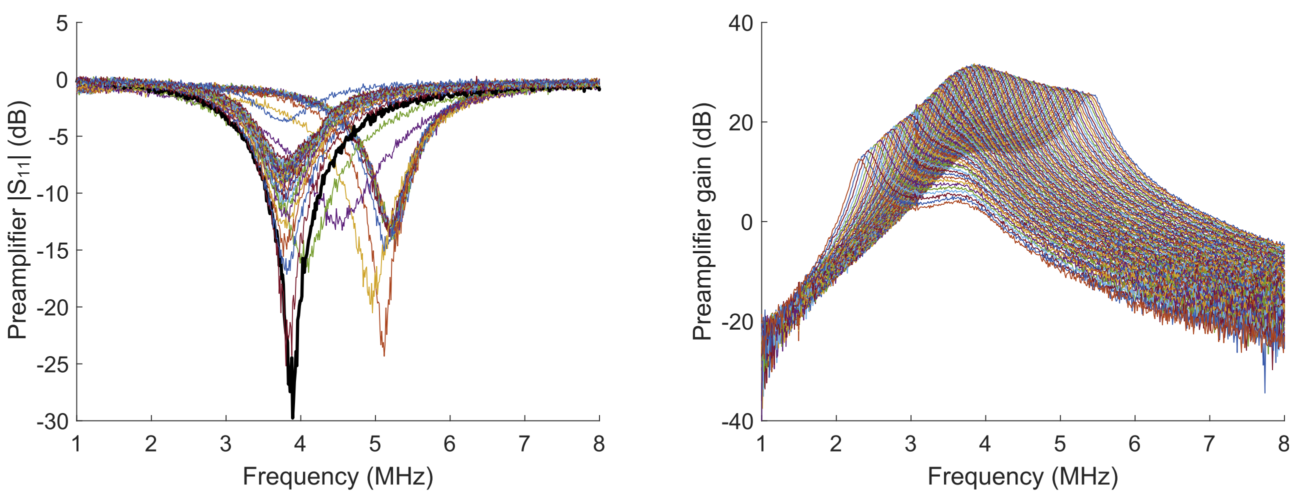

Fig. 7. Measured performance of the preamplifier: (a) reflection coefficient for various gate voltages at a fixed varactor bias voltage; (b) voltage gain for various varactor bias voltages at the fixed gate voltage that corresponds to optimum impedance matching around 3.9 MHz.

impedance matching ($|S_{11}| < -20$ dB) can be obtained over the entire frequency range of interest, i.e., 3.5–5 MHz.

It is important to note that the proposed on-board reflection coefficient measurement technique is not as accurate as using a benchtop instrument. However, in practice it is accurate enough to find the optimum C_1 and C_2 values. For this purpose, the C program first analyzes the estimated $|\Gamma(\omega)|$ curve and then automatically tunes the capacitor arrays in the matching network (i.e., adjusts the values of C_1 and C_2) to minimize $|\Gamma(\omega)|$ at the desired operating frequency.

5.3. Basic NMR experiments

In this section, we describe the results of some basic NMR experiments that were designed to test the capabilities of the spectrometer. All the functions for these experiments were written in C, while the results were either generated on-board (using Python) or on a PC (using MATLAB). Most of the measurements used doped water as the sample; specifically, the water was doped with a paramagnetic compound (copper(II) sulfate) in order to decrease its T_2 and thus the inter-experiment time (in our case, from 2 s to ~400 ms). Optimum values for some of the other parameters used in the experiments were found in advance, i.e., either iteratively or based on simulations. For example, the ratio of excitation to refocusing pulse length in a CPMG sequence (i.e., $t_{p-\pi}/t_{p-\pi/2}$) was set to 1.6, which is known to maximize the signal amplitude in grossly inhomogeneous B_0 fields [35].

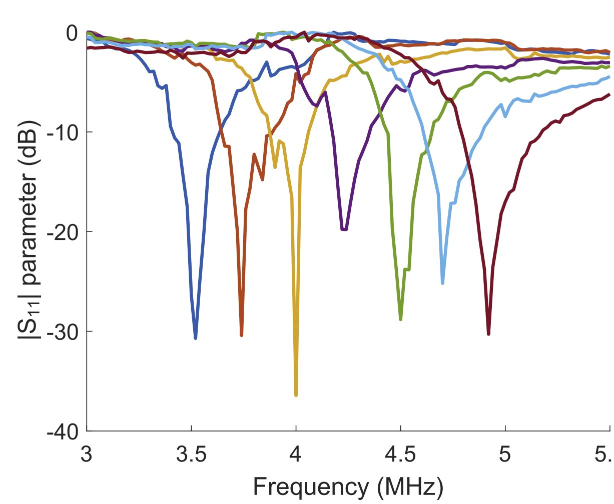

Fig. 8. Measured reflection coefficient of the probe for several combinations of C_1 and C_2. Together, these combinations enable excellent impedance matching $|S_{11}| < -20$ dB to be maintained over the entire 3.5–5 MHz range.

Frequency sweeps are one of the most important functions in NMR experiments, since the signal bandwidth is narrow. The predicted proton Larmor frequency (based on earlier field strength measurements) is first used to tune the matching network and preamplifier, as discussed in the previous section. Then, a set of CPMG experiments is performed at each frequency step over the desired range. The results for a doped water sample are shown in Fig. 9. The maximum signal amplitude is obtained at 4.27 MHz, which is somewhat higher than our predicted Larmor frequency. This result, which will be discussed in more detail later, suggests that the preamplifier and matching network need to be re-tuned to the optimum frequency found from the frequency sweep.

Pulse length sweeps are also frequently needed during NMR experiments. In this example, we first determined the optimum frequency using a frequency sweep, and then varied the excitation and refocusing pulse lengths while keeping the ratio $t_{p-\pi}/t_{p-\pi/2}$ constant at 1.6. Fig. 10 shows the measured signal amplitude versus $t_{p-\pi/2}$. The plot is broadly similar to the ideal sinusoidal nutation curve, but exhibits detailed differences that arise from the grossly inhomogeneous B_0 field within the coil. For example, the first peak in signal amplitude occurs for $t_{p-\pi/2} \approx 11$ μs, while the second occurs near $3 \times 11 = 33$ μs as expected. However, the height of the second peak is only 35% of the first, presumably because the longer pulse has 3× smaller bandwidth. This result shows that the B_0 field is inhomogeneous enough for the effective sample volume to be limited by the amplitude of B_1, i.e., the RF power level.

With the optimum frequency and pulse length parameters now determined, a CPMG experiment was performed to measure T_2 relaxation of the doped water sample. Fig. 11 summarizes the results after averaging across 20 scans. The plot shows that $T_2 \approx 9$ ms and the SNR per echo per scan is ≈ 13. We verified these results by repeating the experiment with the same coil, sample holder, and magnet, but with the rest of the system replaced by a commercial benchtop NMR spectrometer (Kea2, Magritek). Similar values for T_2 and SNR per scan were obtained in the two cases, thus validating the performance of our custom spectrometer.

We have also used our system to analyze more complex samples. For example, quantifying the water content of plant samples is useful in many agricultural and food-related applications. In particular, the water content of flower petals is correlated with freshness, which in turn affects their shelf life. Here we used rose petals (packed tightly within the 5 mm tube) as the sample. The results

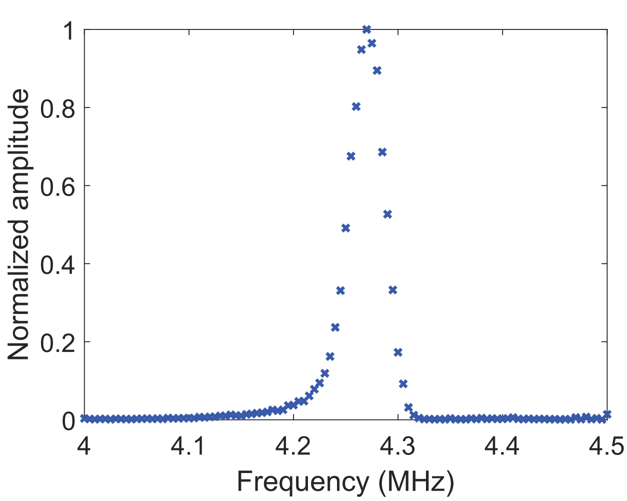

Fig. 9. Measured NMR signal amplitude versus frequency for a sweep between 4 and 4.5 MHz. Experimental settings include temperature = 24.5 °C, $\pi/2$-pulse length = 11 μs, π-pulse length = 17.6 μs, pulse duty cycle = 50%, echo spacing = 300 μs, inter-experiment time = 400 ms, points per echo = 1024, echoes per scan = 128, echo shift = 10 μs, and an averaging factor of 20.

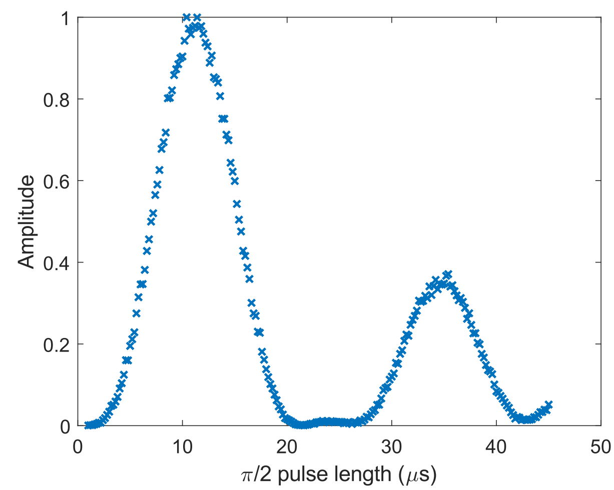

Fig. 10. NMR signal amplitude with pulse length sweep of 1 μs to 45 μs with 0.5 μ spacing. Settings used are temperature of 24.5 °C, B_1 frequency of 4.27 MHz, fixed π-pulse length of 1.6 $\pi/2$-pulse length, pulse duty-cycle of 50%, echo spacing of 300 μs, inter-experiment time of 400 ms, points per echo of 1024, echoes per scan of 128, echo shift of 10 μs, and averaging factor of 20.

from a CPMG experiment are shown in Fig. 12. It can be seen from Fig. 12(d) that the resulting T_2 decay curve is distinctly bi-exponential, unlike for the water sample; this may be due to the presence of distinct "free" and "bound" water components.

Given the second exponential component visible in Fig. 12, it is interesting to perform a 1-D inverse Laplace transform (ILT) to quantify the T_2 distribution. Importantly, the inversion can be performed on the embedded ARM processor, which allows the spectrometer to autonomously optimize NMR acquisition parameters (e.g., inter-experiment time, number of echoes per scan, etc.) based on sample properties. For example, in our earlier work we wrote a C-based 1-D ILT program for the ARM to generate T_1 distributions from inversion recovery data and used it to autonomously optimize the SNR per unit time [36]. Here we use the same program to analyze the rose petal T_2 data. The results, which are shown in Fig. 13, confirm the presence of two T_2 components: one around 7.7 ms and the other (which has larger amplitude) around 86 ms.

We have also used our system for near-zero-field nuclear quadrupole resonance (NQR) experiments. For this purpose, the sample holder was removed from the permanent magnet and placed inside a grounded metal box (i.e., RF shield), while the rest of the setup was unchanged. As an example, Fig. 14 shows the ^{14}N NQR spectrum of sodium nitrite (specifically, the ω_+ line around 4.637 MHz) measured using a spin-locked spin echo (SLSE) pulse sequence [30]. The SLSE sequence is superficially similar to the CPMG sequence except for the fact that $t_{p-\pi}/t_{p-\pi/2} = 1$, and so can be generated by the same pulse program used in earlier experiments.

One of the issues with pulsed measurements such as the SLSE sequence is the limited acquisition time available between pulses. This has the effect of multiplying the true echo signal $s(t)$ with a rectangular window function $w(t)$ of length t_{ACQ}, i.e. the acquired time-domain signal can be written as $s_a(t) = s(t)w(t)$. As a result, the acquired spectrum is the result of convolving the true spectrum with the Fourier transform of the window, i.e., $S_a(\omega) = S(\omega) * W(\omega)$ where '$*$' denotes convolution and $W(\omega)$ is a sinc function. In the case of narrowband signals such as ^{14}N NQR lines, the width of the measured line shape is thus dominated by that of the window, which is about $1/t_{ACQ}$. This effect is visible in the "original" spectrum in Fig. 14. In principle, it can be removed by deconvolution, but the process is numerically ill-conditioned for noisy data. Here we use a regularized deconvolution function to

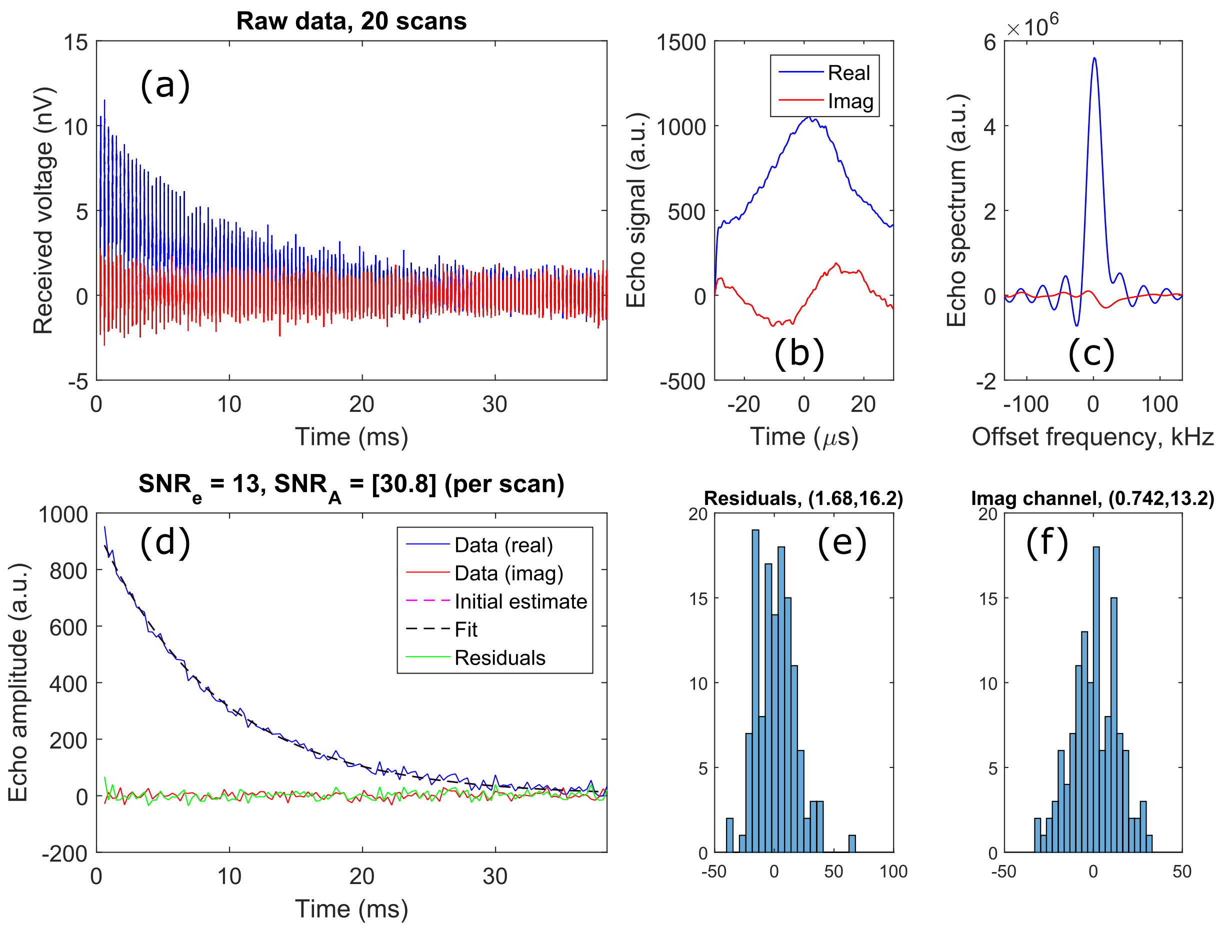

Fig. 11. Measured results for a CPMG experiment on doped water. Experimental settings include frequency = 4.272 MHz, $\pi/2$-pulse length = 11 µs, π-pulse length = 17.6 µs, pulse duty-cycle = 50%, echo spacing = 300 µs, inter-experiment time = 400 ms, points per echo = 1024, echoes per scan = 128, echo shift = 10 µs, and averaging factor = 20. The plots show (a) raw acquired data, (b) average echo shape, (c) average echo spectrum, (d) T_2 decay after matched filtering and mono-exponential fit, (e) histogram of the residuals after fitting for the real channel after matched filtering, and (f) histogram of the imaginary channel after matched filtering.

ensure numerically stable results in the presence of noise. The results (see Fig. 14) show that deconvolution removes most of the side-lobes created by $W(\omega)$, and in this case also reduces the width of the main line by about 2×.

6. Autonomous operation of the spectrometer

6.1. Automated hardware tuning

As discussed, the proposed spectrometer contains a circuit for measuring the reflection coefficient of the matching network. It also includes a digital temperature sensor attached to the permanent magnet. This allows us to implement an embedded temperature sensing and hardware tuning/compensation loop to remove the effect of ambient temperature fluctuations. Note that while the low-noise preamplifier is also tunable, it has not been included in this *direct* feedback loop; this will be explored in future work. Thus, a predefined configuration table is currently used to tune the frequency response of the low-noise preamplifier. The availability of Python programming also enables *indirect* feedback loops in which the NMR signal is monitored in order to optimize the pulse sequence, i.e., find the optimum operating conditions.

There are several cases in which such auto-tuning capabilities may be useful. In the first example, we focus on the problem of temperature variations, which are particularly important for portable spectrometers used in the field, e.g., to analyze agricultural products. Temperature is an environmental variable that is usually not regulated in such field applications. Therefore, it becomes necessary to find the optimum tuning parameters as a function of temperature. Other parameters (e.g., SNR or initial amplitude of the echoes) can also be used as constraints in order to find the optimum spectrometer settings. Fig. 15 shows a flowchart of the algorithm used to find these settings; it runs entirely within the spectrometer and uses measured SNR as the constraint.

The system starts by loading previous settings, which could either be (i) the optimum settings derived from previous measurements, or (ii) derived from initial manual tuning. Then, a CPMG experiment is performed and the SNR of the sample is measured. If the obtained SNR is larger than a threshold, then the system stops as the previous settings are assumed to still be optimum. Otherwise, the system checks the magnet temperature and predicts the Larmor frequency. The matching network is then tuned to the predicted operating frequency by using the reflection measurement feedback loop. At this point, the preamplifier is also tuned using the look-up table method. An NMR frequency sweep operation, based on CPMG experiments, is then performed to find the frequency at which SNR is maximum. The matching network is re-tuned once the optimum Larmor frequency is found. Finally, another CPMG experiment is performed and the SNR is measured. The entire process is repeated until either the constraint is satisfied or the maximum number of iterations is reached.

The results of the auto-tuning experiment are shown in Fig. 16. This experiment is carried out inside a temperature-controlled chamber that allows the ambient temperature to be changed from 0 °C to 70 °C. Fig. 16(a) shows that f_{exp}, the automatically-selected

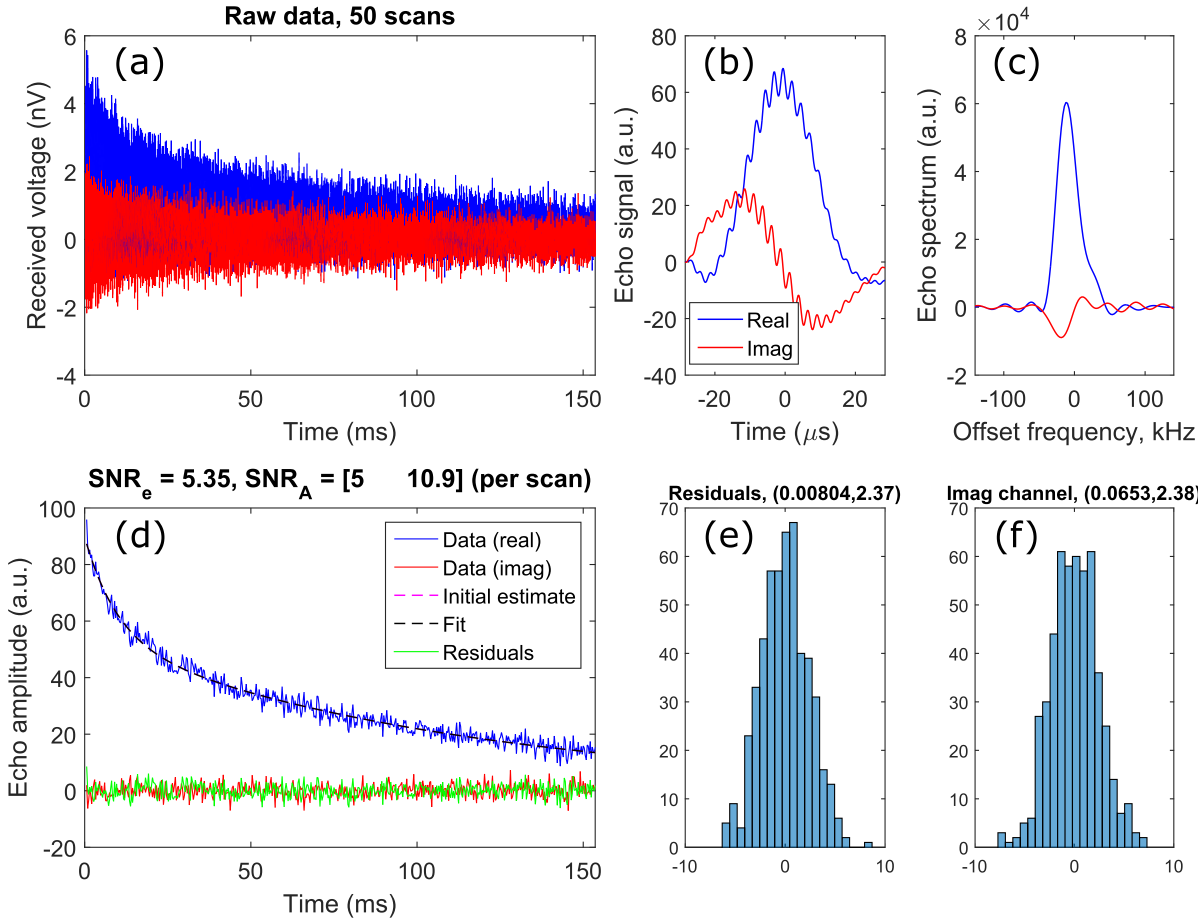

Fig. 12. Measured results for a CPMG experiment on rose petals. Experimental settings include frequency = 4.30 MHz, $\pi/2$-pulse length = 10 µs, π-pulse length = 16 µs, pulse duty-cycle = 50%, echo spacing = 300 µs, inter-experiment time = 1 s, points per echo = 256, echoes per scan = 512, echo shift = 16 µs, and averaging factor = 50. The plots show (a) raw acquired data, (b) average echo shape, (c) average echo spectrum, (d) T_2 decay after matched filtering and bi-exponential fit, (e) histogram of the residuals after fitting for the real channel after matched filtering, and (f) histogram of the imaginary channel after matched filtering.

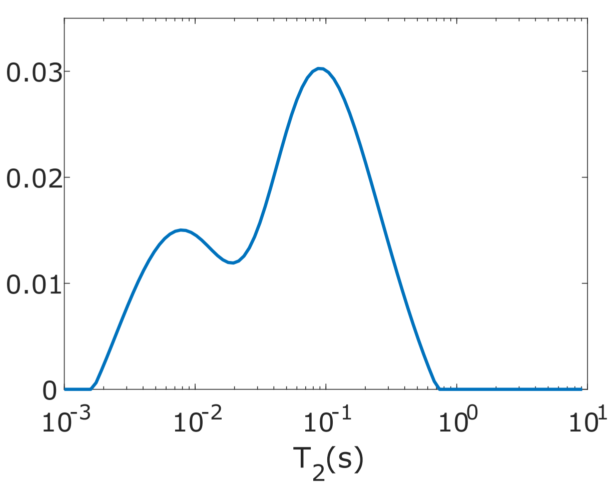

Fig. 13. The normalized T_2 distribution measured for the rose petal, as generated by a C program within the ARM processor. The value of the regularization parameter α was automatically chosen by the program based on the so-called *heel* method [36].

B_1 frequency used for the CPMG scans, closely matches f_{tgt}, the B_1 frequency computed from temperature compensation and the NMR frequency sweep function, as the temperature varies between 5 °C and 65 °C. Moreover, the plot shows that the matching network is automatically tuned such that it is always centered around f_{exp}. Thus, the tuning algorithm ensures that the reflection coefficient around the desired B_1 frequency is always less than −10 dB during the entire experiment. As a result, the SNR of the NMR signal is maintained as the temperature changes. For

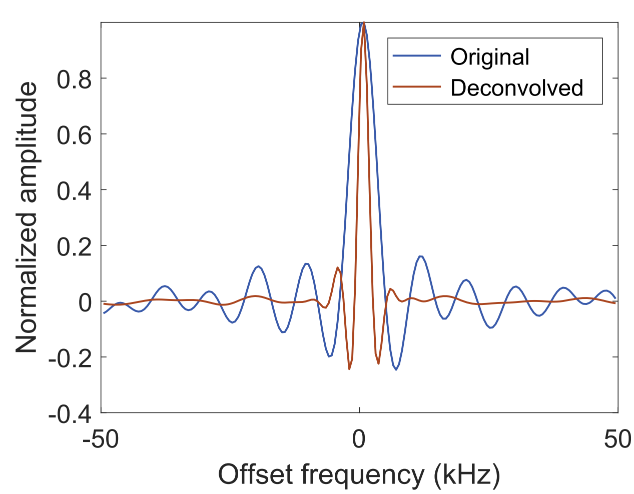

Fig. 14. Measured ^{14}N NQR spectrum obtained from 0.42 gm (0.33 cm^3) of sodium nitrite. Plots are shown with and without regularized deconvolution to remove the effects of the finite-length echo acquisition window. The regularization parameter was automatically determined based on the estimated SNR. Experimental settings include frequency = 4.637 MHz, $\pi/2$-pulse length and π-pulse length = 102.6 µs, pulse duty cycle = 50%, echo spacing = 1000 µs, inter-experiment time = 300 ms, points per echo = 4096, echoes per scan = 32, echo shift = 10 µs, and averaging factor = 512.

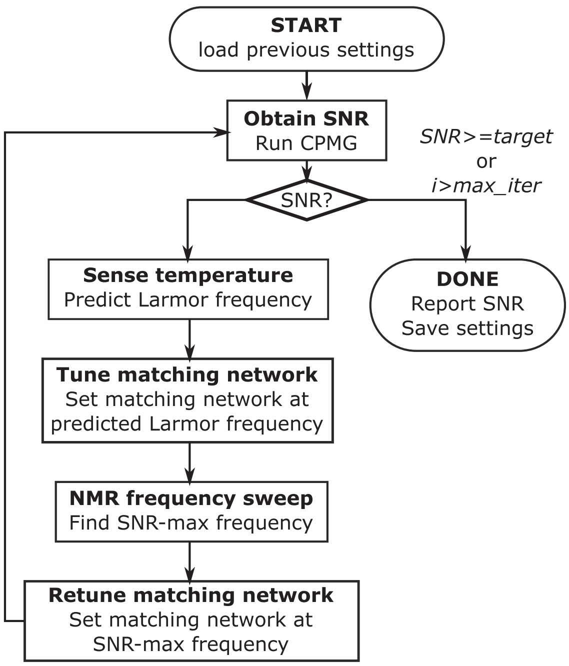

Fig. 15. Flowchart of the proposed auto-tuning algorithm used for temperature compensation.

example, Fig. 16(b) clearly shows that the SNR per echo per scan occasionally decreases below a predefined threshold (10 in this case), but in each ramps back up soon afterwards. This is because each SNR threshold-crossing event triggers the system to automatically re-tune itself in order to compensate for the temperature change.

6.2. Pulse program optimization

Various real-time pulse program optimization methods have been reported in the literature. For example, recent work has utilized stochastic sampling to adaptively adjust sequence parameters (e.g., wait times) in order to minimize the total time required to quantify the properties of individual samples [37]. Here we focus on integrating the various software tuning capabilities that have already been discussed in order to optimize the analysis of different samples. The goal is for the spectrometer to automatically optimize pulse parameters for the current sample in order to maximize SNR per unit time, therefore removing the need of manual tuning. SNR per unit time, denoted by SNR_t, is desired because we not only want the experiment to be effective, but also as short as possible. We can easily derive the total NMR signal strength for a single scan assuming a mono-exponential decay as

$$S_{tot} = \sum_{n=1}^{N_E} a_0 e^{-nT_E/T_2}, \tag{1}$$

where a_0 is the initial signal amplitude, T_E is the echo time, and N_E is the number of echoes. Furthermore, taking longitudinal relaxation T_1 into consideration, the SNR per unit time (in power units) is defined as

$$SNR_t = \frac{{S_{tot}}^2\left(1-e^{-T_W/T_1}\right)^2}{(N_E T_E + T_W)} \frac{1}{N_E \sigma_{noise}^2}, \tag{2}$$

where T_W is the wait time between scans and σ_{noise} is the rms noise per echo per scan (assumed to be additive and white, i.e., uncorrelated between echoes and scans). The equation can be further simplified to

$$SNR_t = \frac{a_0^2}{\sigma_{noise}^2} \frac{1}{T_E} \frac{\left(1-e^{-\alpha}\right)^2\left(1-e^{-\beta}\right)^2}{\beta(\beta+\alpha\delta)}, \tag{3}$$

where we define the dimensionless quantities $\alpha = N_E T_E/T_2$, $\beta = T_W/T_1$, and $\delta = T_1/T_2$. This equation was derived based on the approximation $\frac{1}{1-e^{-x}} \sim \frac{1}{x}$, which is valid when $x = T_E/T_2 \ll 1$. By numerically evaluating non-linear function shown in Eq. (3), we can estimate the optimum pulse sequence parameters over α and β for fixed values of a_0, σ_{noise}, T_E and δ. Fig. 17(a) shows the resulting normalized SNR per unit time for $\delta = 4$. In this case, the optimum values of α and β are 1.53 and 1.05 respectively. Fig. 17(b) plot the optimum values of α and β as

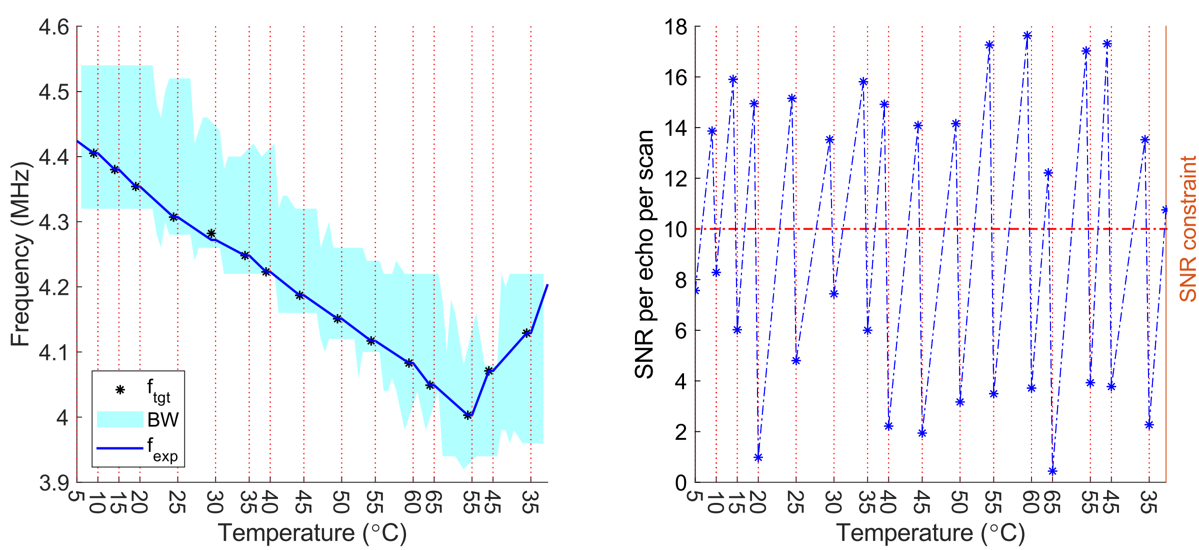

Fig. 16. Experimental results showing that spectrometer auto-tuning can compensate for large temperature fluctuations. Changes in temperature are indicated by vertical striped red lines: during the experiment, the temperature increases from 5 °C to 65 °C and then goes back to 35 °C. Note that the temperature changes do not occur at regular intervals, resulting in variable x-axis separation between adjacent lines. (a) Measured operating frequency of the spectrometer as a function of time. The target frequency (f_{tgt}) is computed via temperature compensation and a CPMG frequency sweep. The B_1 frequency of the pulse sequence (f_{exp}) closely follows f_{tgt}. The −10 dB bandwidth BW of the matching network, denoted as the area shaded in cyan, is also auto-tuned to remain centered around f_{exp}. (b) Measured SNR per echo per scan for CPMG scans as the temperature changes. The horizontal red striped line is the SNR constraint for the experiment, which is set to 10 in this case. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

a function of δ. As δ (i.e., the $T1/T2$ ratio) increases, it becomes beneficial to acquire more echoes while reducing the repetition time.

An algorithm was developed to automatically maximize the SNR per unit time as described above. A flowchart of the algorithm is shown in Fig. 19. Because the samples may not have mono-exponential decays, it performs 1-D Laplace inversions to account for all T_1 and T_2 relaxation components; in each case, the regularization parameter is set using the *heel* method. We first start by using a CPMG experiment with a fixed T_E to optimize the number of echoes needed to give us the maximum SNR. In this first scan, we collect enough echoes to ensure that all T_2 components decay by at least two time constants, thus ensuring an accurate Laplace inversion. We then extract the longest significant T_2 component from the inversion and use this value to optimize the number of echoes N_E. By selecting the longest component, we guarantee that all data is collected in future scans. However, the optimum value of $\beta = N_E T_E / T_2$ depends on $\alpha = T_W / T_1$ (see Fig. 17), which is unknown since we haven't acquired any T_1 information yet. Thus, we temporarily select the value of N_E that results in $\beta = 1$. This is a reasonable estimate based on the plot, and will be improved once T_1 inversion is performed.

We now conduct an inversion recovery (IR) experiment using the value of N_E estimated above. We set the longest wait time to a value of $10 \times T_2$, which is more than sufficient to make sure that all T_1 relaxation components can be detected. As explained above, the IR data is inverted to extract all T_1 components. The longest significant component is extracted from the data and used for T_W optimization. Now, since $\delta = T_1/T_2$ is known, the optimum values of α and β can be read from Fig. 17. Thus, we have localized the sample within the feasible region of the (T_1, T_2) plane, which is denoted by the shaded region in Fig. 18. The optimal values of both N_E and T_W can now be determined. Note that this optimization process has avoided the use of 2-D Laplace inversions, which are computationally intensive on the embedded ARM processor.

Once the pulse parameters are optimized, the system runs CPMG experiments continuously to monitor the contents of the sample coil. Here we assume that relevant changes in sample properties can be detected directly from the echo decays, so we avoid inverting the data during the monitoring period to reduce total processing time. Instead, we compute the sums of the first and second halves of the echo train, which are denoted by a_1 and a_2 respectively. This process allows us to improve SNR through accumulation while retaining both amplitude and T_2 information. We start the monitoring process by collecting 10 separate a_1 and a_2 data points and computing their mean and standard deviation. The a_1 and a_2 values measured for the next CPMG data set are then compared to their statistics from the previous 10 scans. The condition for determining if the sample has not changed is

$$\left|\overline{a_1} - a_1'\right| \leqslant 3\sigma_{a_1} \quad \text{and} \quad \left|\overline{a_2} - a_2'\right| \leqslant 3\sigma_{a_2}, \tag{4}$$

where σ_{a_1} and σ_{a_2} are the standard deviations of the a values respectively, $\bar{a_1}$ and $\bar{a_2}$ are their means, and a_1' and a_2' are the newly collected data points. If Eqn. 4 is satisfied, the system will replace the oldest a_1 and a_2 data points with a_1' and a_2' and continue to monitor the current sample. Similarly, the condition for determining if the sample has changed is

$$\left|\overline{a_1} - a_1'\right| > 3\sigma_{a_1} \quad \text{or} \quad \left|\overline{a_2} - a_2'\right| > 3\sigma_{a_2}. \tag{5}$$

If Eq. 5 is satisfied, the system will re-optimize parameters for the newly detected sample and continue the process.

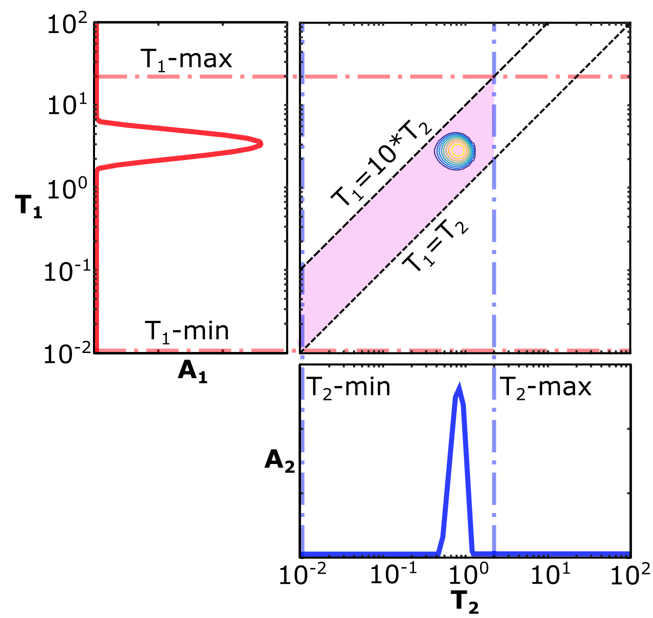

Fig. 18. An example of the measured T_1, T_2, and joint (T_1, T_2) distributions of a test sample (in this case, milk). Colors in the 2-D plot denote constant magnitude contours of the (T_1, T_2) distribution, while T_1 and T_2 axes are in sec. The feasible region of (T_1, T_2) for an unknown sample, assuming $1 \leqslant \delta \leqslant 10$, is denoted by the shaded region. The boundaries of this region (i.e., $T_{1,min}$, $T_{1,max}$, $T_{2,min}$, and $T_{2,max}$) are determined by prior knowledge. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

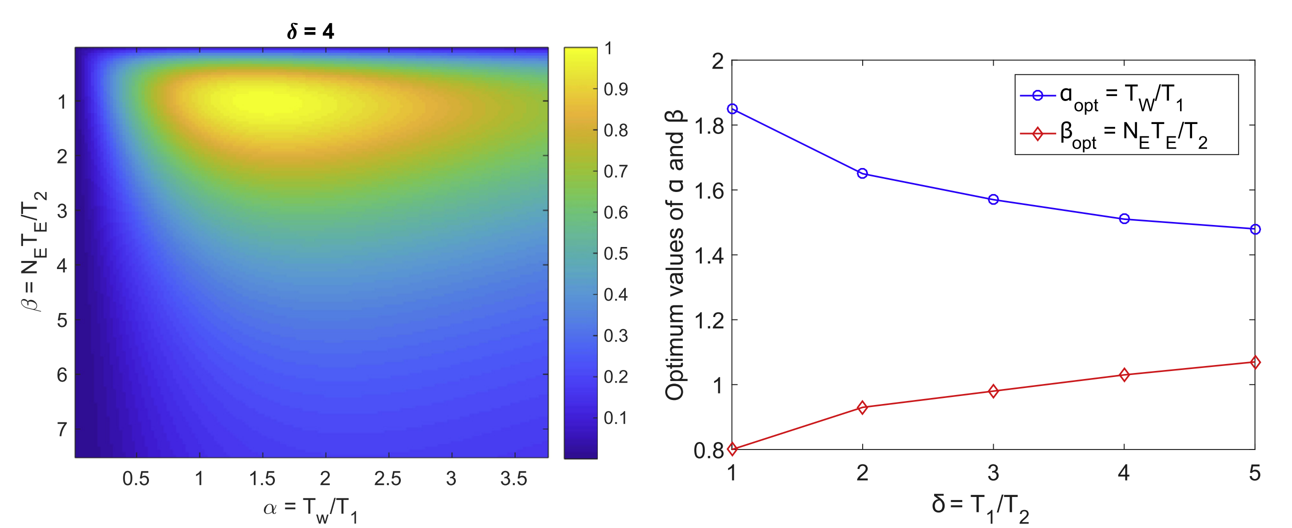

Fig. 17. (a) Theoretical dependence of SNR per unit time on α and β for $\delta = 4$. (b) Theoretical dependence of the optimum values of α and β as a function of δ.

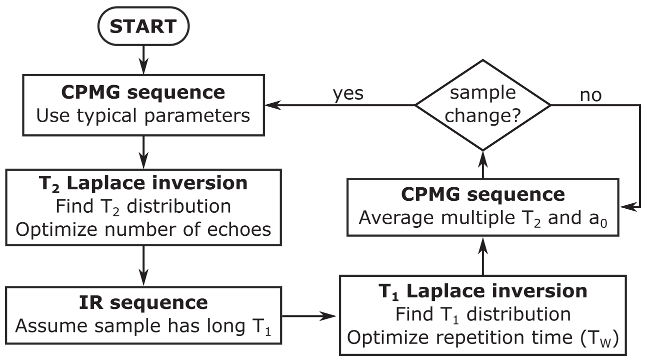

Fig. 19. Flowchart of the proposed auto-tuning algorithm used for sample change detection and parameter optimization.

The proposed algorithm was implemented and experimentally verified with 3 different samples, i.e. milk, mineral oil, and doped water, which were randomly inserted into the spectrometer. The result is shown in Fig. 20. The system is able to quickly detect when the sample changes by performing the routine shown in Fig. 19 to automatically obtain optimized NMR pulse parameters. Experimentally, the slowest T_2 components found for milk, mineral oil, and doped water were approximately 225 ms, 48 ms, and 10 ms respectively; the optimum values of N_E were 2259, 480, and 102 respectively; the slowest T_1 components were 1.26 s, 146 ms, and 12 ms, respectively; and the optimum values of T_W were 2 s, 235 ms, and 19 ms respectively.

7. Conclusion

We have described an autonomous, highly-portable NMR spectrometer based on a low-cost SoC that integrates an FPGA fabric with an embedded ARM processor. The ability of the ARM processor to run a full Linux OS enables embedded C programming for low-level processing, as well as embedded Python programming for high-level processing. For example, a C program that implements the 1-D ILT has been developed for generating T_1 and T_2 distributions directly within the spectrometer. A highly-tunable AFE that interfaces with the SoC has also been developed, thus realizing a complete and self-contained instrument. The benefits of the proposed hardware and software approach have been experimentally verified by performing basic NMR experiments on a variety of samples. Moreover, we have demonstrated the ability of the system to auto-tune itself to compensate for large temperature changes, i.e., satisfy a user-defined constraint such as minimum SNR per scan. We have also demonstrated the ability of the system to auto-tune pulse sequence parameters after changes in sample properties.

In future work, we will move the quadrature down-conversion operation from Python to the FPGA fabric in order to allow the acquired data to be downsampled before it is stored in the ADC FIFO. This will allow the same FIFO to hold data from much longer scans, i.e., acquire more echoes per scan. In addition, we will increase the frequency tuning range of the preamplifier to allow operation over a wider range of field strengths. Finally, we will also increase the peak RF output power level of the transmitter to support larger sample sizes.

Acknowledgments

The authors would like to thank students in the Case Western Reserve University course EECS 326/600 "Measurements and Instrumentation" for testing and debugging several early versions of the proposed spectrometer. This work was funded by the National Science Foundation (NSF) through grant CCF-1563688.

Appendix A. Power consumption comparison

Table A.1 below compares the typical power consumption of a normal personal computer (PC), laptop, the Terasic DE1-SoC, and a SoM (Critical Link MitySOM-5CSx) that uses the same chip as the DE1-SoC. The DE1-SoC already requires far less power (7.4 W while making NMR measurements and processing data) compared to a PC (> 100 W) or a laptop (30–90 W). However, the DE1-SoC board contains several unnecessary peripherals that consume additional power. The MitySOM-5CSx is added as a comparison point since it is a small form-factor SoM that does not include unnecessary peripherals. Thus, such SoMs are good candidates

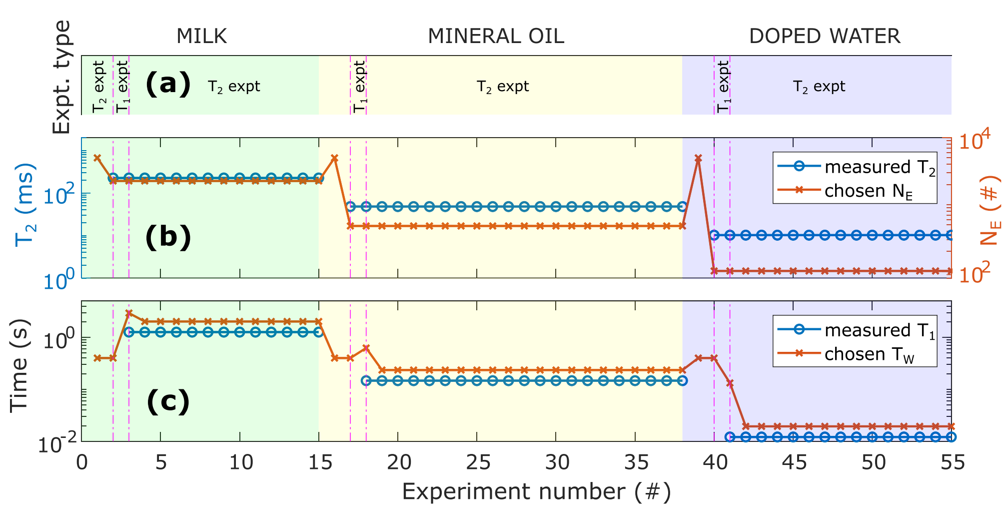

Fig. 20. Results of an automatic parameter optimization experiment on three samples (milk, mineral oil, and doped water). Different colors denote the sample used at any given time. Changes in experiment type are denoted by striped vertical red lines. (a) Experiment type selected by the instrument. (b) Measured slowest T_2 component obtained using T_2 Laplace inversion along with corresponding number of echoes (N_E) used in the experiment. It can be seen that N_E varies between samples according to the measured T_2. The rise in N_E at the beginning and when the sample is changed corresponds to loading typical parameters in the program flowchart, which eventually converge to a value proportional to the sample T_2. (c) Measured slowest T_1 component obtained using T_1 Laplace inversion along with corresponding repetition time (T_W) used in the experiment. After every sample change, T_W starts with typical values and eventually converges to a value proportional to the sample T_1. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Table A.1
Power consumption comparison table.

Device	Power
Desktop PC	> 100 W
Laptop	30–90 W
Terasic DE1-SoC	6.7 W (idle), 7.4 W (load)
Critical Link MitySOM-5CSx	2.05 W (idle, clean FPGA fabric)

for realizing miniaturized and ultra-low-power spectrometers with similar measurement capabilities as the current system. Note that this SoM uses ~2.05 W in an idle state, i.e., with no program running on the FPGA fabric [38]. We predict that the power consumed by the SoM in an active state will lie between 2.05 W and 7.4 W.

Appendix B. Hardware details

B.1. Transmitter design

The transmitter converts a small input signal to a linearly-polarized sinusoidal RF magnetic field within the sample coil. This field can be decomposed into the sum of two equal-amplitude circularly-polarized components rotating in opposite directions, one of which (generally denoted by B_{1c}) perturbs the nuclear spins. It is desirable to maximize the power efficiency of the transmitter by delivering as much as possible of the available RF power to the coil, i.e., minimizing the power reflected back to the source due to impedance mismatches. In our case, the input is directly driven by two FPGA pins that generate differential square-wave signals. This allows us to avoid the use of a high-speed DAC, but does introduce the problem of harmonic content. In particular, a square wave of period $T = 2\pi/\omega_0$ and amplitude V_0 can be expanded into the Fourier series

$$v(t) = \frac{4V_0}{\pi} \sum_{n=1,3,5\ldots}^{\infty} \frac{1}{n} \cos(n\omega_0 t), \tag{B.1}$$

which has harmonics at odd multiples of ω_0 that decay slowly (as $1/n$) with frequency. Typical lumped matching networks are only matched at the fundamental frequency ω_0. For example, the input impedance of the common two-capacitor network is capacitive at the harmonics, so all harmonic power is reflected back into the transmitter. In most cases this power is then dissipated by dissipative elements within the transmitter (transistors or protection circuits), resulting in a loss of efficiency. A distributed matching network based on transmission lines can avoid this problem, but is bulky at low frequencies. Instead, we remove most of the harmonic content using a low-pass filter before it is delivered to the coil. However, this approach requires the filter to be programmable; in particular, while ω_0 can vary between experiments, the cutoff frequency must always lie between ω_0 and $3\omega_0$ to remove all the harmonics.

Our transmitter design uses a custom class-A push-pull linear power amplifier with differential inputs and single-ended output, as shown in Fig. B.21. A class-A topology was chosen due to its linearity, which keeps the total harmonic distortion (THD) of the output waveform low while allowing its amplitude to be varied using the duty cycle of the input square waves generated by the FPGA. This control procedure is also known as pulse-width modulation (PWM). In our case, the FPGA generates two differential PWM input signals, i.e., with a time delay of $T/2$ (half a period) between them, in order to minimize even-harmonic distortion and external common-mode noise. Defining $0 < D < 1$ as the duty cycle, the differential waveform can be written as

$$v(t) = \frac{4V_0}{\pi} \sum_{n=1,3,5\ldots}^{\infty} \frac{\sin(n\pi D)}{n} \cos(n\omega_0 t), \tag{B.2}$$

where $D = 0.5$ corresponds to the special case of the square wave, which was analyzed in Eqn. (B.1). Using differential input signals preserves the symmetry of the waveform, which cancels out all the even harmonics. Moreover, the amplitude of the fundamental component is equal to $(4V_0/\pi) \times \sin(\pi D)$, i.e., can be smoothly varied from 0 to a maximum of $4V_0/\pi$ by increasing D from 0 to 0.5.

In our implementation, the two PWM inputs are buffered with BJT-based (BC846, Nexperia) emitter followers and then passed through a differential low-pass filter to remove harmonics. The latter uses a multiple-feedback active filter topology based on a fully-differential high-speed op-amp (THS4141, Texas Instruments). Note that the cutoff frequency of this filter is fixed at ~6 MHz in this design, which is adequate for the limited expected operating range of $\omega_0/(2\pi) = 3.5$–4.5 MHz. However, it will need to be made programmable in order to operate over a broader range, as mentioned earlier.

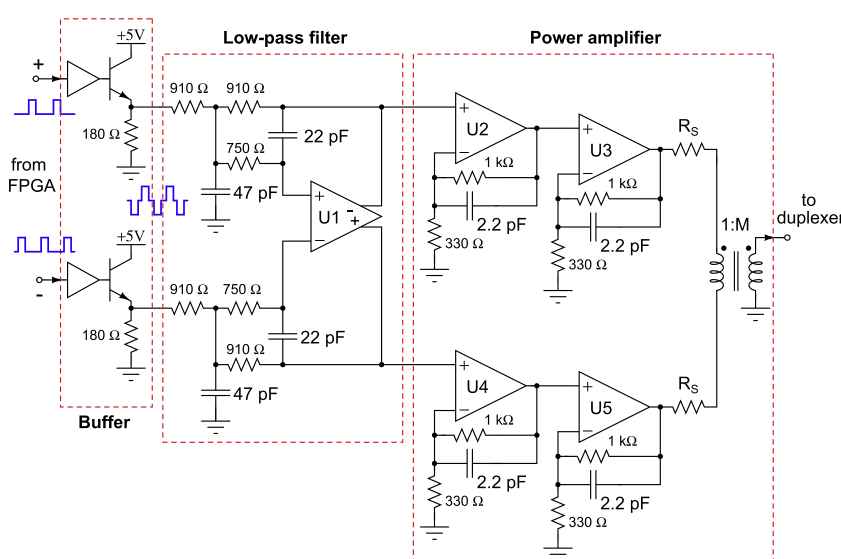

Fig. B.21. The NMR transmitter, which consists of a differential buffer followed by a multiple-feedback differential active low-pass filter, two-stage power amplifier, and output transformer.

The filter is followed by a two-stage differential power amplifier based on high-speed, high-output current op-amps (LM7372, Texas Instruments). The second stage of power amplifier (U3 and U5) can be switched manually between LM7372 and LT1210 (Analog Devices), with a trade-off between higher-speed lower-output current and lower-speed higher-output current, respectively. These op-amps are chosen to simplify the design while maximizing the available output power. The maximum bandwidth of this transmitter design is certainly limited by the LM7372 or LT1210 to 40 MHz or 11.6 MHz for a gain of 3, respectively. However, without the programmable low-pass filter mentioned earlier, the system is currently not able to operate at its maximum bandwidth. The differential outputs of the amplifier are converted to single-ended form by a RF transformer with a 1:M turns ratio. The transformer also scales the output impedance of the amplifier by M^2 to reach the desired value of 50 Ω that minimizes the reflection coefficient. Estimating the optimal value of M is not trivial since the second-stage op-amps behave as nonlinear large-signal circuits that do not have a constant and well-defined output impedance R_{out}. However, we can usefully approximate R_{out} as the ratio between (i) the voltage drop from the supply voltage V_{DD} to the peak output voltage V_P, and (ii) the peak output current I_P, i.e., as

$$R_{out} \approx \frac{V_{DD} - V_P}{I_P}. \tag{B.3}$$

Simulation of the LM7372 as the final amplifier stage using a freely-available circuit simulator (LTspice, Analog Devices) shows $V_P = 11.6\,\text{V}$ for $V_{DD} = 15\,\text{V}$ and $I_P = 150\,\text{mA}$, resulting in $R_{out} \approx 22.7\,\Omega$. We should add $2R_s$ to this value if optional series resistors are added to the op-amp outputs (see Fig. B.21); here we assume $R_s = 0$. In this case a step-up transformer with $M = 1.5$ (Coilcraft PWB series) provides an output impedance of ~51 Ω, i.e., nearly the correct value. The resulting maximum output power level available from a ±15 V power supply is measured to be approximately 2.5 W. The same calculation can be performed for LT1210, which requires step-down transformer with $M = 1/2$ for output impedance matching, to obtain approximately 10.6 W of maximum output power from a ±15 V power supply.

B.2. Duplexer design

The transmitter generates large voltages across the sample coil that can potentially damage the receiver preamplifier. To isolate the receiver from the transmitter during transmit mode, and vice versa, a transmit-receive switch (duplexer) is used. The duplexer also acts as a damper to minimize coil ringing after transmit pulses. Our design is based on the well-known passive lumped duplexer shown in Fig. B.22, which was originally proposed by McLachlan [29].

In *transmit mode*, the transmitter output is large enough to turn on either $D1$ or $D2$ (depending on signal polarity), allowing it to drive the matching network. In addition, either $D3$ or $D4$ also turn on, so the receiver path resembles a parallel LC circuit (formed by L_1 and C_1) that provides a high impedance at resonance to protect the preamplifier. Further protection is provided by $D5$ and $D6$, which turn on to short the preamplifier input terminal to ground. In *receive mode*, the NMR signal induced on the sample coil is not large enough to turn on any of the diodes. This effectively disconnects the transmitter from the matching network. Moreover, the receiver path resembles a series LC circuit (formed by L_1 and C_2) that is designed to be resonant around the operating frequency. As a result, it provides a low impedance path for the received signal to reach the preamplifier.

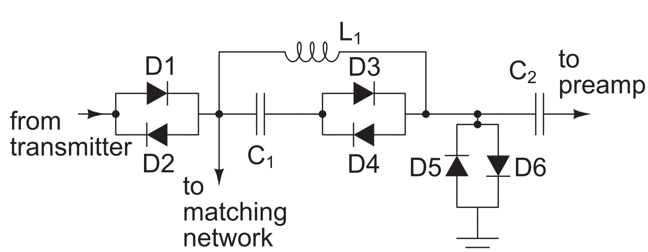

Fig. B.22. Passive duplexer circuit used in the proposed analog front-end.

Ideally, the duplexer should be broadband enough to cover the entire operating frequency range without tuning. However, the proposed design relies on series or parallel LC circuits in the receiver path, which provide optimum performance (e.g., maximum isolation in transmit mode, minimum attenuation in receive mode) near the corresponding resonant frequencies of $1/\sqrt{L_1C_1}$ and $1/\sqrt{L_1C_2}$, respectively. Fortunately, both these tuned circuits are heavily loaded by either the transmitter output impedance or the receiver input impedance (both nominally 50 Ω), which lowers their quality factor (Q) and provides sufficient bandwidth. Moreover, this also eliminates the need to choose high-Q capacitors and inductors for implementing the duplexer.

B.3. Tunable matching network design

A passive impedance-matching network is used to transform the impedance of the coil to a real value of $Z_0 = 50\,\Omega$ near the Larmor frequency. This ensures maximum power transfer from the transmitter to the coil in transmit mode, and from the coil to the preamplifier in receive mode. We use a two-capacitor network in which the shunt and series capacitors are denoted by C_1 and C_2, respectively. The input impedance Z_{in} seen by the duplexer is then given by

$$\begin{aligned} Z_{in} &= \frac{(j\omega L + R_p)\frac{1}{j\omega C_{1,eff}}}{j\omega L + R_p + \frac{1}{j\omega C_{1,eff}}} + \frac{1}{j\omega C_2} \\ &= \underbrace{\frac{j\omega L + R_p}{(1 - \omega^2 LC_{1,eff}) + j\omega R_p C_{1,eff}}}_{Z_1} - \underbrace{\frac{j}{\omega C_2}}_{Z_2}, \end{aligned} \tag{B.4}$$

where $C_{1,eff}$ is defined as the sum of C_1 and the coil parasitic capacitance (C_p), R_p is defined as the coil parasitic resistance, and L defined as the coil inductance. Intuitively, C_1 resonates with the coil and its value is set such that the resonant frequency $1/\sqrt{LC_{1,eff}}$ is slightly higher than the Larmor frequency ω_0. As a result, $\Re(Z_{in})$ can be set to 50 Ω near ω_0. The series capacitor C_2 then cancels out the imaginary part of Z_{in}, i.e., ensures $\Im(Z_{in}(\omega_0)) = 0$. In practice, C_1 and C_2 are chosen to be high-Q mica capacitors to minimize losses due to their parasitic resistance.

However, the network is narrowband and only provides good matching over a bandwidth of approximately $2\omega_0/Q$, where Q is the quality factor of the coil near ω_0. Thus, at the relatively low frequencies (3.5–4.5 MHz) targeted in this work, a typical value of $Q = 80$ results in a bandwidth of only ~100 kHz, which is not sufficient to handle changes in B_0 due to manufacturing tolerances, temperature fluctuations, etc. [31]. We use a digitally-tunable two-capacitor matching network to avoid this issue. The design used in this paper is based on the work of Chen et al. [30], and is shown in Fig. B.23. The circuit uses a bank of low-loss switches (reed relays from Coto Technologies) to set the values of C_1 and C_2. To simplify computations, each capacitor is designed to closely approximate a 8-bit binary-weighted array. In particular, C_1 and C_2 are chosen to have least-significant bit (LSB) values of 18 pF and 2 pF, respectively. This allows C_1 to be adjusted in 18 pF steps up to a maximum of ~4.7 nF, and C_2 to be adjusted in 2 pF steps up to ~630 pF.

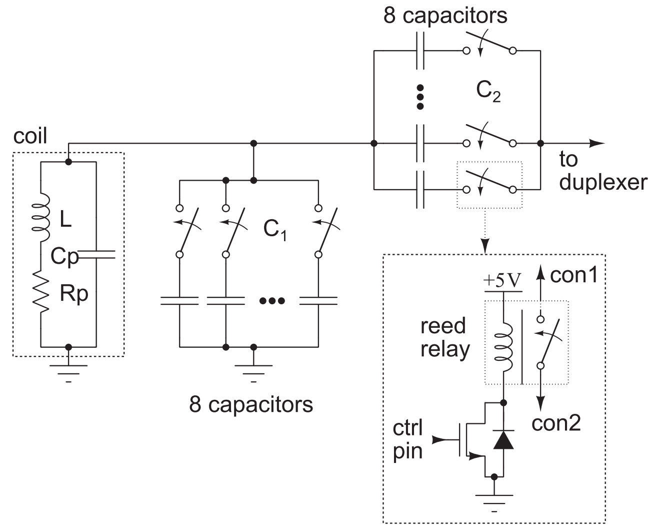

Fig. B.23. The proposed tunable two-capacitor matching network, which uses binary-weighted 8-bit capacitor arrays for both C_1 and C_2. The switches used to set the capacitor values are reed relays with drivers that are shown in the inset on the bottom right.

B.4. Reflected power measurement

In practice, the coil and matching network contain parasitic capacitors that are difficult to model accurately. As a result, significant errors often occur between the predicted and actual matching frequency. Accurate tuning therefore requires the frequency response of the matching network to be directly measured. The reflection coefficient $\Gamma(\omega) = S_{11}$ is a suitable performance parameter in this case, since $|\Gamma(\omega)|^2$ and $1 - |\Gamma(\omega)|^2$ quantify the fractional RF power that is reflected from the network and delivered to the coil, respectively. We have implemented a one-port embedded network analyzer to measure $|\Gamma(\omega)|$, i.e., perform the "wobble" function. The circuit, which is shown in Fig. B.24, uses a directional coupler (SYDC-20-61VHP+, Mini-Circuits) to measure the reflected power level with a coupling factor of −20 dB. The voltage developed across the load resistor $R_1 = 50\ \Omega$ is then buffered and amplified by high-speed op-amps (LT6230, Analog Devices) before being digitized. The same ADC (LTC1746, Analog Devices) is reused to digitize the NMR signal, so an analog switch is included on the board to select between the NMR signal and reflected power measurement paths.

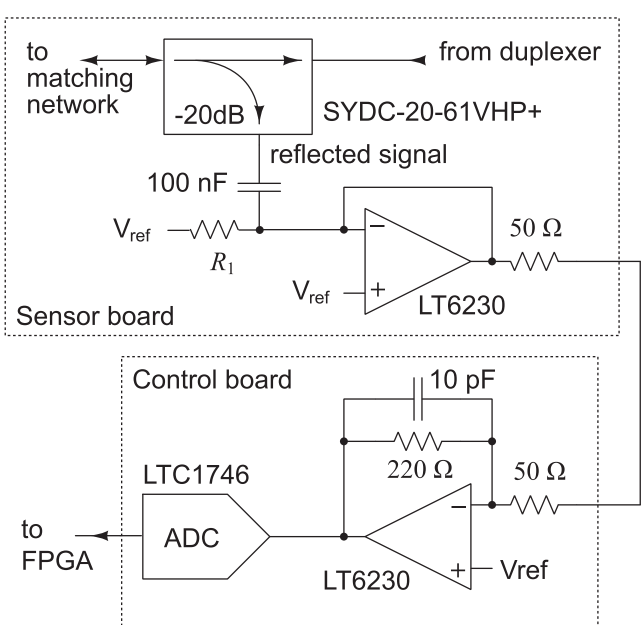

Fig. B.24. The reflection measurement circuit. The reflected power is measured with a −20 dB directional coupler and buffered with an op-amp before being amplified and digitized.

B.5. Tunable preamplifier and second-stage amplifier design

As the NMR signal is narrow-band, it is intuitive to use a narrow-band preamplifier to minimize noise figure (NF) and unwanted interference from external sources. In our previous work [31], a preamplifier with a narrow-band input matching network and tuned output load was designed to achieve this goal. While this topology works well in simulation, in practice it is difficult to control since both the input and output networks have to be tuned. Moreover, these networks interact with each other through the parasitic capacitance of the transistors, which makes tuning a complicated iterative process. In this design, we simplify the tuning problem by using a common-gate preamplifier that does not require an input matching network. In fact, the circuit naturally provides both a broadband resistive input impedance and a moderate amount of voltage gain.

The proposed circuit is shown in Fig. B.25. The first stage uses a common-gate topology implemented using a low-noise JFET (J310, ON Semiconductor). By itself, this results in an intrinsically broadband resistive input impedance of $Z_{in} \approx 1/g_m$, where g_m is the transconductance of the JFET. Moreover, g_m (and thus Z_{in}) can be controlled by changing the gate bias voltage.

However, to obtain $Z_{in} = 50\ \Omega$ impedance, we need a relatively high value of $g_m = 20$ mS, which leads to high power consumption and also degrades the NF. An inverting step-up transformer $TF1$ with a turns-ratio of 1:N_1 is added to the design between the gate and source terminals of the JFET to reduce the required value of g_m, which also reduces NF. In particular, it can be shown that

$$NF \approx 1 + \frac{\gamma g_m}{R_s(N_1+1)^2 g_m^2} = 1 + \frac{\gamma}{N_1+1}, \qquad \text{(B.5)}$$

where $\gamma \approx 2/3$ is known as the excess noise factor of the JFET and the approximation is valid if the voltage gain of the circuit is $\gg 1$. In addition, the second expression is valid when $(N_1+1)g_m R_s = 1$ to realize input impedance matching. As a result, the required value of $g_m = 1/((N_1+1)R_s)$ decreases by a factor of (N_1+1), which allows the bias current to be significantly reduced (thus saving

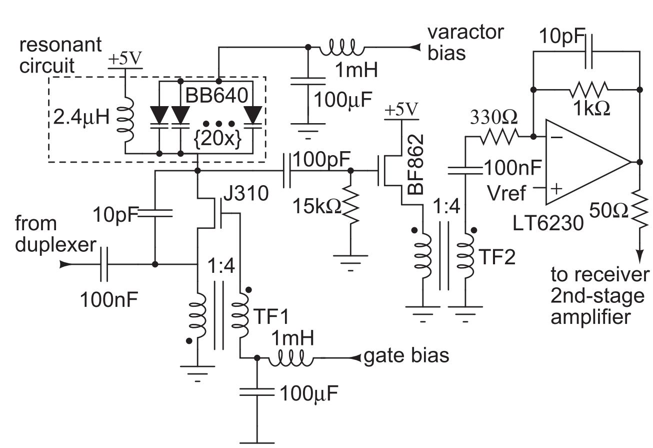

Fig. B.25. Schematic of the proposed tunable preamplifier. The input impedance of the first-stage preamplifier is tunable using gate bias voltage, while its frequency response is tunable using varactor bias voltage. Both these voltages are controlled by a DAC. The amplified signal is buffered and sent to the second-stage amplifier.

power) and also reduces the noise figure. Fundamentally, this is because *TF*1 amplifies the input signal by a factor of $(N_1 + 1)$ *without adding noise*. In this case $N_1 = 4$ (using a Coilcraft PWB series transformer), thus reducing NF from 2.2 dB (for $N_1 = 0$) to 0.55 dB.

At the drain of the input JFET, twenty varactors (BB640, Infineon Technologies) in parallel with a fixed inductor ($L_f = 2.4\ \mu$H) are used to realize a tunable *LC* resonator. The varactor bias voltage is digitally controlled from 0 to −10 V via a DAC (TCA9555PWR, Texas Instruments), thus providing a bandpass frequency response whose center frequency can be tuned by the FPGA. Any noise in the DAC output voltage is removed by a second-order *LC* low-pass filter with a low cutoff-frequency of ~500 Hz. The filtered output of the first stage is further amplified by the next stage, which provides a broadband gain of $N_2 = 4$. This circuit consists of a JFET-based source follower buffer (BF862, NXP Semiconductors) followed by a 1:N_2 step-up transformer (*TF*2). Finally, a high speed op-amp (LT6230, Analog Devices) provides additional gain and a 50 Ω output impedance suitable for driving a coaxial cable.

The overall small-signal voltage gain A_v of the preamplifier is given by

$$A_v(\omega) = \frac{1}{2} g_{m1}(N_1+1)(N_2)\left(\frac{(j\omega L_f + R_f)\frac{1}{j\omega C_{var}}}{j\omega L_f + R_f + \frac{1}{j\omega C_{var}}}\right), \tag{B.6}$$

where g_{m1}, N_1, N_2, and L_f are defined as before, C_{var} is the varactor capacitance, R_f is the series resistance of L_f, and the $\frac{1}{2}$ factor arises from impedance matching. The frequency-dependent term in the brackets is simply a second-order bandpass response (Lorentzian) with a peak magnitude of $R_f Q^2$ (where $Q = \omega L_f / R_f$ is the quality factor of L_f), center frequency of $\omega_0 = 1/\sqrt{L_f C_{var}}$, and a bandwidth of ω_0/Q. In our case C_{var} decreases by about 5× as the reverse bias voltage increases from 0 to −10 V, so ω_0 can be controlled over a range of about 1:$\sqrt{5}$, i.e., more than an octave.

A second-stage amplifier consisting of four op-amp gain stages provides more signal amplification after the preamplifier, as shown in Fig. B.26. The gain of the last stage can be varied over a range of 26 dB by using analog switches, thus increasing the overall dynamic range of the receiver. These switches are controlled by digital signals from the FPGA, thus allowing the overall gain to be configurable from 40 to 66 dB.

B.6. Analog-to-digital conversion

In previous work [31], we used a 14-bit, 4.5 Msps serial ADC (LTC2314, Analog Devices) to digitize the receiver outputs. The serial topology makes it easy to interface with the FPGA, since the digital outputs are encoded on a standard three-wire serial peripheral interface (SPI) bus. However, the serial topology increases the bit rate on this bus to a minimum of (bits per sample) × (sampling rate), i.e., 14 × 4.5 = 63 Mbps in this case. In practice the data rate was even higher (87.5 Mbps) due to additional clock cycles required during every conversion period. SPI buses are not designed to carry such high data rates; in particular, they are not impedance matched, which results in significant signal integrity issues and bit error rates whenever bus lengths exceed a few cm. The design of the FPGA-based ADC controller also became challenging since it had to run at twice the bit rate, i.e., at 175 MHz.

Finally, even with such high bit rates, the original ADC could only sample at 4.5 Msps; this is the fastest value commercially available with SPI outputs but is still lower than the Nyquist rate for the NMR signals of interest, which occur in the 3.5–5 MHz range. Thus, a bandpass-sampling technique [39] was used to post-process the acquired data. This method works since individual NMR scans generate relatively narrowband signals (typical bandwidth < 100 kHz). However, certain sampling rates are not allowed in order to avoid aliasing; moreover, the sampled output spectrum is in general not centered around zero frequency (DC). Calculating the resulting offset frequency requires exact knowledge of both the ADC sampling rate and the Larmor frequency. The latter is set by a local oscillator (LO) generated by a phase-locked loop (PLL) on the FPGA, but in general drifts with respect to the clock used by the ADC. As a result, successive scans have slightly different offset frequencies and have to be individually phase-rotated before they can be phase-cycled and averaged, which becomes challenging when the SNR per scan is low.

The fundamental source of these problems was the choice of an ADC with a *serial* output bus. In the current design, we therefore switched to a 14-bit, high-speed (25 Msps) ADC (LTC1746, Analog Devices) with a *parallel* output bus. Due to the parallel topology, the FPGA control logic can run at the ADC sample rate, i.e. 25 MHz, which greatly simplifies the design by relaxing its timing constraints. Moreover, the new sampling rate satisfies the Nyquist criterion over the 3–5 MHz operating range, so the output signal can be directly down-converted to DC without the need for bandpass sampling. If a higher-speed ADC is required for some applications, the ADC can be upgraded to a compatible 80 Msps design (LTC1748, Analog Devices) from the same family.

Appendix C. The CPMG pulse sequence

Referring to Fig. C.27, labels shown in black, i.e., the $\pi/2$ pulse length ($t_{P-\pi/2}$), the π pulse length ($t_{p-\pi}$), and echo spacing (t_{ECHO}), are to be specified by the user within the C program in human-readable format, e.g. in input variables that have units of μs. These values are then converted to the number of clock cycles by the C program before being stored into the NMR control register in the FPGA. The ADC acquisition window (t_{ACQ}) is computed as the number of samples per echo (*SpE*) multiplied by the clock frequency. Here *SpE* is a parameter specified by the user and is directly stored

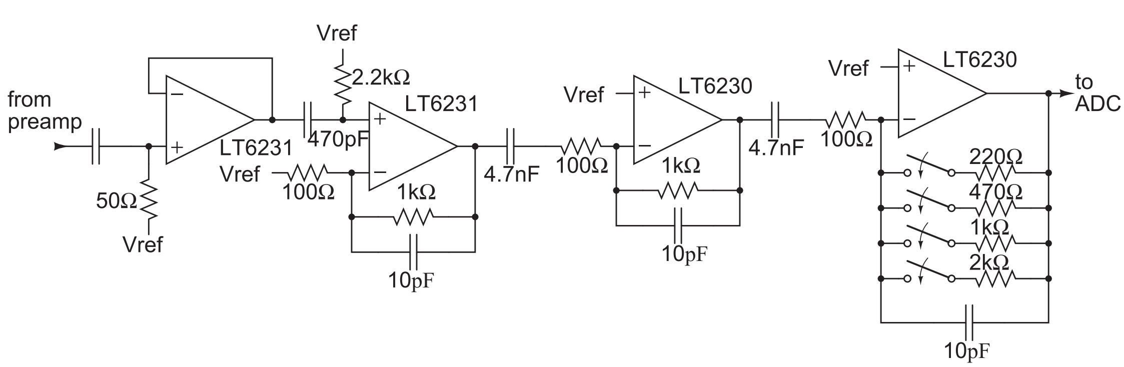

Fig. B.26. The second stage amplifier, which consists of four op-amp gain stages. The gain of the last stage is digitally configurable using switches.

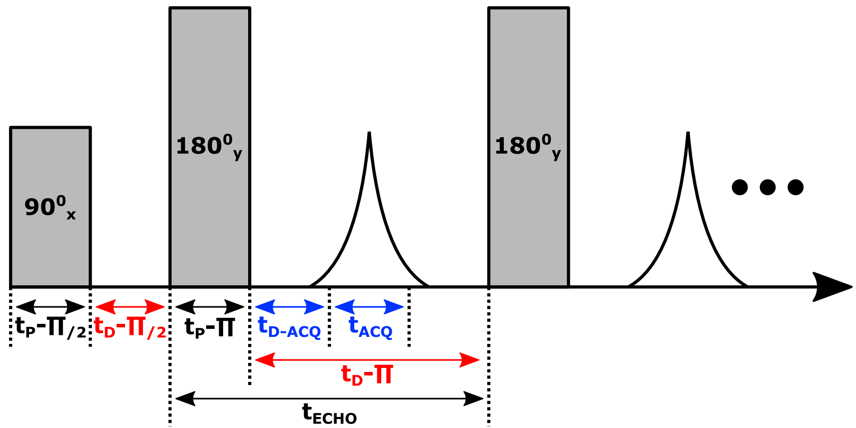

Fig. C.27. Generalized CPMG pulse sequence implemented on the spectrometer. The $\pi/2$ pulse is generated once, followed by multiple π pulses. The black labels denote user-specified variables within the C program, and the red labels denote variables that are computed by the ARM processor. The blue labels shows variables that are related to the ADC, which uses a different clock frequency, with t_{ACQ} also being specified by the user in the C program.

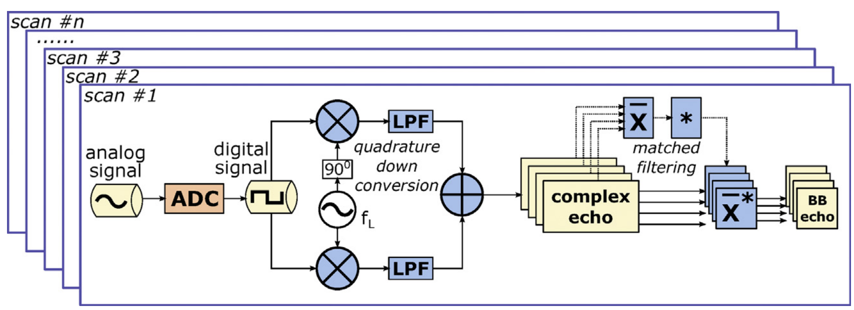

Fig. D.28. Basic technique used for processing NMR signals, including digitization, quadrature down-conversion, and matched filtering. All digital signal processing is performed in Python.

in the NMR control register. All other parameters, i.e. delay after the $\pi/2$ pulse ($t_{\mathrm{D}-\pi/2}$), delay after the π pulse ($t_{\mathrm{D}-\pi}$), and delay before ADC acquisition ($t_{\mathrm{D-ACQ}}$) are derived by the ARM processor using the following equations:

$$\begin{aligned} t_{\mathrm{D}-\pi/2} &= \frac{1}{2}\, t_{ECHO} - t_{\mathrm{P}-\pi/2}, \\ t_{\mathrm{D}-\pi} &= t_{ECHO} - t_{\mathrm{P}-\pi}, \\ t_{\mathrm{D-ACQ}} &= \frac{1}{2}\, t_{ECHO} - t_{\mathrm{P}-\pi}. \end{aligned} \tag{C.1}$$

Appendix D. NMR signal processing with python

Fig. D.28 summarizes the basic signal processing technique used to obtain baseband echo from the analog signal, as performed using Python. The digitized RF data is first down-converted to baseband using a quadrature down-converter and then low-pass filtered to the desired bandwidth. In order to simplify the computations, multiplication by square wave signals is used for down-conversion. Moreover, given that the ADC sampling rate is $4\times$ the RF frequency, each period of these square waves can be simplified to the sequence $\{+1, 0, -1, 0\}$, which further simplifies the implementation.

The complex down-converted signal, which consists of a series of echoes, is then matched-filtered by the conjugate of the average of all acquired echoes. This generates a single complex number for each echo; these numbers are then phase-rotated to ensure that the signal (echo amplitudes) only appears in one of the two components (usually the real component), while the other component contains only noise. Note that the average of all echoes used here for matched filtering can be viewed as the best-available estimate of the true (i.e., noiseless) echo shape. Therefore, the overall operation effectively weighs the noisy acquired signal by its estimated shape, which can be shown to be the optimal estimation procedure in the presence of additive white noise [30]. The resulting echo amplitudes can be further processed to obtain relaxation and diffusion time constants.

References

[1] M.D. Hürlimann, Well logging, in: R.K. Harris, R.L. Wasylishen (Eds.), eMagRes, American Cancer Society, 2012. doi:https://doi.org/10.1002/9780470034590.emrstm0593.pub2.
[2] J. Perlo, E. Danieli, J. Perlo, B. Blümich, F. Casanova, Optimized slim-line logging NMR tool to measure soil moisture in situ, J. Magn. Reson. 233 (2013) 74–79.
[3] B. Blümich, F. Casanova, M. Dabrowski, E. Danieli, L. Evertz, A. Haber, M. Van Landeghem, S. Haber-Pohlmeier, A. Olaru, J. Perlo, et al., Small-scale instrumentation for nuclear magnetic resonance of porous media, New J. Phys. 13 (1) (2011) 015003.
[4] H. Lee, E. Sun, D. Ham, R. Weissleder, Chip–NMR biosensor for detection and molecular analysis of cells, Nat. Med. 14 (8) (2008) 869.
[5] H.-Y. Chen, Y. Kim, P. Nath, C. Hilty, An ultra-low cost NMR device with arbitrary pulse programming, J. Magn. Reson. 255 (2015) 100–105, https://doi.org/10.1016/j.jmr.2015.02.011.
[6] X. Liang, S. Binghe, M. Yueping, Z. Ruyan, A digital magnetic resonance imaging spectrometer using digital signal processor and field programmable gate array, Rev. Sci. Instrum. 84 (5) (2013) 054702, https://doi.org/10.1063/1.4803007.

[7] W. Tang, W. Wang, A single-board NMR spectrometer based on a software defined radio architecture, Meas. Sci. Technol. 22 (1) (2011) 015902.
[8] J. Yun, J. Yu, T. Hongyan, L. Gengying, A complete digital radio-frequency source for nuclear magnetic resonance spectroscopy, Rev. Sci. Instrum. 73 (9) (2002) 3329–3331, https://doi.org/10.1063/1.1497500.
[9] K. Takeda, A highly integrated FPGA-based nuclear magnetic resonance spectrometer, Rev. Sci. Instrum. 78 (3) (2007) 033103, https://doi.org/10.1063/1.2712940.
[10] S. Jie, X. Qin, L. Ying, L. Gengying, Home-built magnetic resonance imaging system (0.3 T) with a complete digital spectrometer, Rev. Sci. Instrum. 76 (10) (2005) 105101, https://doi.org/10.1063/1.2069707.
[11] W.K. Peng, L. Chen, J. Han, Development of miniaturized, portable magnetic resonance relaxometry system for point-of-care medical diagnosis, Rev. Sci. Instrum. 83 (9) (2012) 095115, https://doi.org/10.1063/1.4754296.
[12] L. Gengying, J. Yu, Y. Xiaolong, J. Yun, Digital nuclear magnetic resonance spectrometer, Rev. Sci. Instrum. 72 (12) (2001) 4460–4463, https://doi.org/10.1063/1.1386629.
[13] C.A. Michal, K. Broughton, E. Hansen, A high performance digital receiver for home-built nuclear magnetic resonance spectrometers, Rev. Sci. Instrum. 73 (2) (2002) 453–458, https://doi.org/10.1063/1.1433950.
[14] S.B. Belmonte, R.S. Sarthour, I.S. Oliveira, A.P. Guimarães, A field-programmable gate-array-based high-resolution pulse programmer, Meas. Sci. Technol. 14 (1) (2003) N1.
[15] S. Handa, T. Domalain, K. Kose, Single-chip pulse programmer for magnetic resonance imaging using a 32-bit microcontroller, Rev. Sci. Instrum. 78 (8) (2007) 084705, https://doi.org/10.1063/1.2773636.
[16] C.A. Michal, A low-cost spectrometer for NMR measurements in the earth's magnetic field, Meas. Sci. Technol. 21 (10) (2010) 105902.
[17] D. Ha, J. Paulsen, N. Sun, Y.-Q. Song, D. Ham, Scalable NMR spectroscopy with semiconductor chips, Proc. Natl. Acad. Sci. 111 (33) (2014) 11955–11960, https://doi.org/10.1073/pnas.1402015111.
[18] W. Wolf, A.A. Jerraya, G. Martin, Multiprocessor system-on-chip (MPSoC) technology, IEEE Trans. Comput.-Aid. Des. Integr. Circ. Syst. 27 (10) (2008) 1701–1713.
[19] L.H. Crockett, R.A. Elliot, M.A. Enderwitz, R.W. Stewart, The Zynq Book: Embedded Processing with the ARM Cortex-A9 on the Xilinx Zynq-7000 All-Programmable SoC, Strathclyde Academic Media, 2014.
[20] L.A. Colnago, F.D. Andrade, A.A. Souza, R.B. Azeredo, A.A. Lima, L.M. Cerioni, T. M. Osán, D.J. Pusiol, Why is inline nmr rarely used as industrial sensor? Challenges and opportunities, Chem. Eng. Technol. 37 (2) (2014) 191–203.
[21] T.D. Beer, A. Burggraeve, M. Fonteyne, L. Saerens, J. Remon, C. Vervaet, Near infrared and raman spectroscopy for the in-process monitoring of pharmaceutical production processes, Int. J. Pharmaceut. 417 (1) (2011) 32–47, https://doi.org/10.1016/j.ijpharm.2010.12.012, Advanced characterization technique <http://www.sciencedirect.com/science/article/pii/S0378517310009324> .
[22] C.M. Preston, Applications of nmr to soil organic matter analysis: history and prospects, Soil Sci. 161 (1996) 144–166, https://doi.org/10.1097/00010694-199603000-00002.
[23] E.T. Engman, Applications of microwave remote sensing of soil moisture for water resources and agriculture, Rem. Sens. Environ. 35 (2) (1991) 213–226, https://doi.org/10.1016/0034-4257(91)90013-V. <http://www.sciencedirect.com/science/article/pii/003442579190013V> .
[24] W.M. Post, K.C. Kwon, Soil carbon sequestration and land-use change: processes and potential, Global Change Biology 6(3), 317–327 (Available from: arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1046/j.1365-2486.2000.00308.x). doi:https://doi.org/10.1046/j.1365-2486.2000.00308.x. <https://onlinelibrary.wiley.com/doi/abs/10.1046/j.1365-2486.2000.00308.x>.
[25] F. Djodjic, K. Börling, L. Bergström, Phosphorus leaching in relation to soil type and soil phosphorus content, J. Environ. Qual. 33 (2) (2004) 678–684.
[26] F. Qu, Z. Zhang, J. Hu, J. Xu, S. Wang, Y. Wu, Adaptive dual-sensor noise cancellation method for continuous wave mud pulse telemetry, J. Petrol. Sci. Eng. 162 (2018) 386–393, https://doi.org/10.1016/j.petrol.2017.12.058, <http://www.sciencedirect.com/science/article/pii/S092041051731015X>.
[27] S.M. Mwachaka, A. Wu, Q. Fu, A review of mud pulse telemetry signal impairments modeling and suppression methods, J. Petrol. Explor. Prod. Technol. doi:https://doi.org/10.1007/s13202-018-0483-y.
[28] S.J. Stolfo, M.B. Salem, A.D. Keromytis, Fog computing: mitigating insider data theft attacks in the cloud, in: 2012 IEEE Symposium on Security and Privacy Workshops, 2012, pp. 125–128. doi:https://doi.org/10.1109/SPW.2012.19.
[29] L. McLachlan, Lumped circuit duplexer for a pulsed NMR spectrometer, J. Magn. Reson. (1969) 39 (1) (1980) 11–15, https://doi.org/10.1016/0022-2364(80)90152-3.
[30] C. Chen, F. Zhang, S. Bhunia, S. Mandal, Broadband quantitative NQR for authentication of vitamins and dietary supplements, J. Magn. Reson. 278 (2017) 67–79, https://doi.org/10.1016/j.jmr.2017.03.011.
[31] D.J. Ariando, A portable low-cost NMR spectrometer, MS thesis, Case Western Reserve University, January 2018.
[32] H.Y. Carr, E.M. Purcell, Effects of diffusion on free precession in nuclear magnetic resonance experiments, Phys. Rev. 94 (3) (1954) 630.
[33] S. Meiboom, D. Gill, Modified spin-echo method for measuring nuclear relaxation times, Rev. Sci. Instrum. 29 (8) (1958) 688–691.
[34] S. Mandal, S. Oh, M.D. Hrlimann, Absolute phase effects on CPMG-type pulse sequences, J. Magn. Reson. 261 (2015) 121–132, https://doi.org/10.1016/j.jmr.2015.10.014.
[35] M. Hürlimann, D. Griffin, Spin dynamics of Carr–Purcell–Meiboom–Gill-like sequences in grossly inhomogeneous B0 and B1 fields and application to NMR well logging, J. Magn. Reson. 143 (1) (2000) 120–135.
[36] C. Chen, A. Srivastav, D. Ariando, S. Mandal, R. Tang, Y.-Q. Song, Real-time data inversion methods for low-field nuclear magnetic resonance (NMR), in: 23rd International Conference on Digital Signal Processing (DSP 2018), 2018.
[37] Y.-Q. Song, Y. Tang, M. Hürlimann, D. Cory, Real-time optimization of nuclear magnetic resonance experiments, J. Magn. Reson. 289 (2018) 72–78.
[38] Critical Link, MitySOM-5CSx System on Module (11 2014). <https://www.criticallink.com/wp-content/uploads/MitySOM-5CSx-Datasheet.pdf>.
[39] R.G. Vaughan, N.L. Scott, D.R. White, The theory of bandpass sampling, IEEE Trans. Signal Process. 39 (9) (1991) 1973–1984, https://doi.org/10.1109/78.134430.