Received: 5 September 2018 | Revised: 19 November 2018 | Accepted: 26 November 2018

DOI: 10.1002/cmr.b.21401

ORIGINAL ARTICLE

WILEY

A low-cost multi-channel software-defined radio-based NMR spectrometer and ultra-affordable digital pulse programmer

Carl A. Michal[1,2]

[1]Department of Physics and Astronomy, University of British Columbia, Vancouver, British Columbia, Canada

[2]Department of Chemistry, University of British Columbia, Vancouver, British Columbia, Canada

Correspondence
Carl A. Michal, Department of Physics and Astronomy, University of British Columbia, Vancouver, BC, Canada.
Email: michal@physics.ubc.ca

Funding information
Natural Sciences and Engineering Research Council of Canada, Grant/Award Number: RGPIN-2014-05114

Abstract

In recent years, a new generation of software-defined radio (SDR) products has emerged that are well suited for adaptation for NMR instrumentation. The software and hardware of a new NMR spectrometer console based upon an inexpensive SDR product are described. This device is provided with open-source firmware and drivers which have allowed the customization of its behavior and optimization for use in NMR. In particular, these low-level modifications have allowed for straightforward synchronization of multiple SDR boards with each other and with other devices. Control of other devices is facilitated by a new digital pulse programmer constructed using an inexpensive open-source, widely available microcontroller platform. The strategy used for precise, reproducible synchronization of the different components is described in detail. Representative spectra are presented demonstrating both high-resolution and wide-line spectroscopy with spectral widths up to 50 MHz.

KEYWORDS
arduino, double-resonance, LimeSDR, quadrature receiver, vector modulator

1 | INTRODUCTION

The rapid development of wireless communication technologies over the past couple of decades has been a boon for the NMR spectrometer builder. Ever more capable components continue to become available at progressively lower cost. One trend in the wireless industry has been the development of software-defined radio (SDR) systems. An SDR is a radio system that implements modulation/demodulation capabilities that have previously been implemented in hardware, in software. Some examples are transmitters and receivers for wireless computer networks, digital television, global positioning systems, and cellular telephone signals where special purpose hardware devices for modulating or demodulating data onto rf signals are being replaced by flexible radio systems where the function is determined by software.

In recent years, several publications have described the development of low-cost NMR systems (eg, 1-5), NMR systems built upon SDR devices,[6-9] and custom integrated circuits for NMR.[10-13] Many of these systems are limited to low-field use, and the vast majority have been single-frequency instruments. For a multi-channel NMR spectrometer capable of double, triple, or quadruple resonance experiments at low or high field, precise, reproducible synchronization of multiple channels is necessary, as is precisely timed switching of other functions such as amplifier blanking and receiver gating. While some commercial SDR platforms do supply a suitable synchronization capability for multiple rf channels, lower-cost, easier to use SDR platforms generally do not.

Recently a new, wholly open-source SDR device has come onto the market that provides a compelling option for NMR. The LimeSDR[14] is a small (12 × 7 cm), inexpensive (US$299), USB3 interfaced circuit board capable of rf transmit and receive over a frequency range of 100 kHz to 3.8 GHz. Both transmit and receive channels are equipped with programmable-gain amplifiers and sophisticated digital signal processing modules, allowing the board to function as a nearly complete NMR spectrometer console. The hybrid analog/digital nature of the rf on this device allows for elimination of the artifacts associated with analog rf systems while simultaneously covering the full range of relevant frequencies. The open-source nature of the software and firmware supporting the device has allowed a number of modifications to be made to allow predictable timing of multiple devices to nanosecond precision. A key feature of the present work is the development of the synchronization mechanisms needed to reproducibly synchronize multiple rf channels with each other, and with externally produced logic signals.

For control of external gates, amplifier blanking, and other devices, we have developed an ultra low-cost, expandable pulse programmer that provides 25 digital logic outputs and can be synchronized precisely and reproducibly with the rf outputs of the LimeSDR boards. The entire spectrometer system is driven by home-built software that provides a graphical user interface along with a powerful but simple C-based pulse programming language, along with basic NMR data processing functionality.

In this article, we begin with a brief description of an idealized analog transmitter and receiver system before describing the hardware and software of the present spectrometer. This is followed by details of the requirements and strategy for synchronizing multiple devices to sufficient precision. Finally, we demonstrate the new system's capabilities on both high-resolution and broad-line NMR.

2 | HARDWARE DESIGN

2.1 | Idealized analog transmitter and receiver

The rf sections of an idealized single-frequency analog transmitter (TX) and receiver (RX) for NMR are shown in Figure 1. The transmitter incorporates stages for phase modulation, frequency shifting, amplitude modulation, gain control, and finally an rf switch to gate the signal. The TX output is at a frequency given by the difference between the local oscillator (LO) and intermediate frequency (IF) signals. Phase modulation is accomplished by varying the amplitudes of in-phase (I) and quadrature (Q) IF signals, according to the outputs of the I & Q digital to analog convertors (DACs). The two quadrature IF signals are then combined to produce a signal at the IF frequency with the chosen phase. This structure is known as a vector modulator, or I & Q modulator, and can be implemented as a single device or may be built out of discrete mixers, splitters, and combiners. The phase-modulated IF signal is then frequency shifted by mixing with the LO signal to produce the final transmit RF frequency. We use the label RF to refer to the final frequency, RF = LO − IF, while rf is an abbreviation for radio-frequency. The use of an IF frequency for phase modulation that is then shifted by mixing with a LO is usually necessary because the components that provide the phase-shifted IF signals for the vector modulator tend to have narrow useful bandwidths. Broadband rf performance is achieved by varying the LO frequency, but keeping the IF frequency constant. The mixer that shifts the IF to RF can be either a single-sideband mixer which produces only the desired difference frequency, or might be an ordinary double-balanced mixer that produces both LO + IF and LO − IF, in which case some filtering of the mixer output to remove the undesired sum frequency might be necessary. Following the shift to the final RF frequency, the transmitter incorporates a linear amplitude modulation according to a third DAC, followed by a variable-gain amplifier/attenuator allowing the rf amplitude to be adjusted over a large (logarithmic) range before a final rf gate.

On the receive side, the incoming NMR signals are first gated with an rf switch to protect the downstream components of the receiver from preamplifier overload during rf pulses. A variable-gain amplifier is incorporated to adjust the signal amplitude before mixing with the LO to frequency shift from RF to IF. This signal is then split into two, and mixed with the in-phase and quadrature IF signals to produce I & Q “baseband” signals. These signals are then low-pass filtered and sampled by analog to digital convertors (ADCs). This structure of mixing the incoming signals with in-phase and quadrature references is known as quadrature detection, and allows signals at frequencies above and below the RF frequency to be distinguished. In practical implementations of transmitters and receivers such as these, the programmable-gain amplifiers would likely be implemented as multiple fixed-gain amplifiers and programmable attenuators distributed at different points through the transmit and receive chains. Additional filtering might also be included.

A major drawback of systems such as those in Figure 1 is that good performance depends on accurate matching of pairs of components such as mixers, filters, DACs, and ADCs, on an accurate 90° phase shift in the quadrature reference of the IF, and accurate amplitude balance between the in-phase and quadrature IF signals. The most severe effects of minor imperfections in any of these arise on the receive side, where false “image” signals can be generated that appear with the opposite offset of the true signal with respect to the RF frequency. Phase cycling is commonly used to remove these kinds of artifacts, by averaging the signals over both of the I & Q channels in the receiver on successive experiments.[15] In modern digital systems, these component matching and balancing requirements have been eliminated by performing all of the quadrature operations in the digital domain.

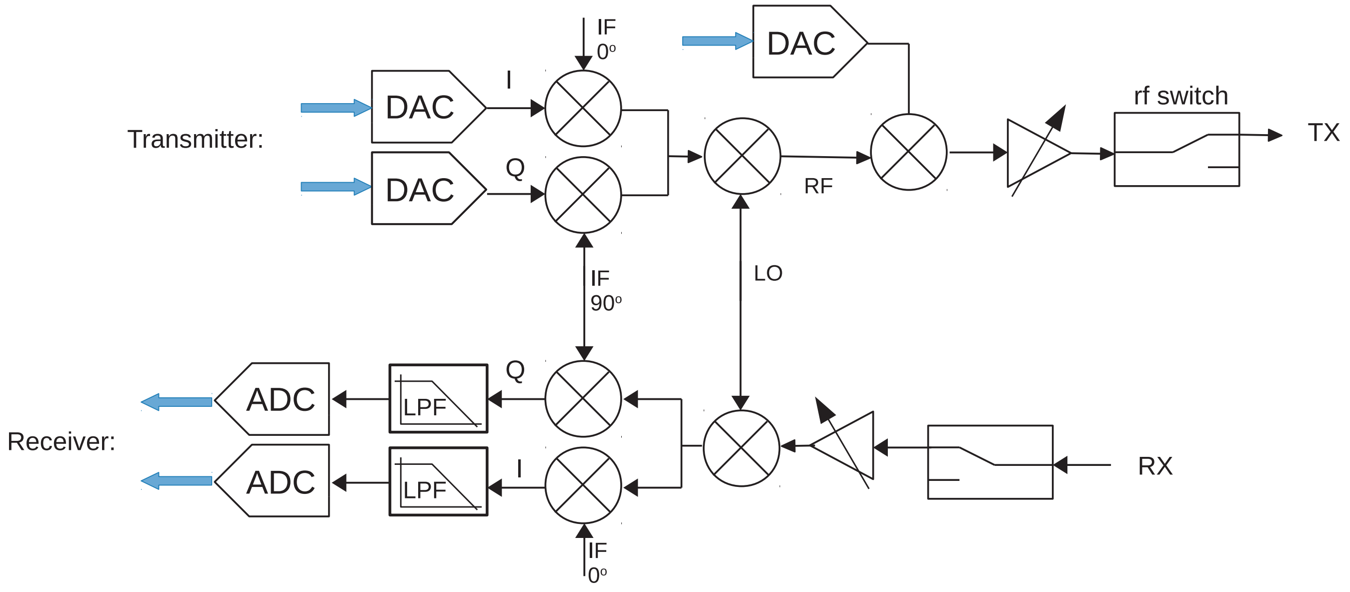


FIGURE 1 An idealized analog NMR spectrometer transmitter and receiver. Digital data to control the digital to analog convertors (DACs) must be supplied by a digital pulse programmer or other controller, while received digital data from the analog to digital convertors (ADCs) are transmitted to a host computer. Thick arrows represent the flow of digital data to DACs for transmit or from ADCs on receive

2.2 | LimeSDR Internals

The LimeSDR board, which provides the rf functionality for the new spectrometer, contains three main functional devices: a Cypress FX3 USB3 interface chip (Cypress Semiconductor Corp, San Jose, CA), an Altera Cyclone IV-E FPGA (Intel Programmable Solutions, San Jose, CA), and an LMS7002M rf chip (Lime Microsystems, Guildford, UK). In addition, there are a number of supporting devices, including 256 MB of DDR2 SDRAM, and several chips facilitating the clocking of the main components.

A highly simplified diagram of the internal design of the LMS7002M is shown in Figure 2. The transmitters consist of vector modulators followed by programmable-gain output amplifiers, while the receivers are analog quadrature detectors following programmable-gain amplifiers. Both transmit and receive channels incorporate programmable bandwidth analog low-pass filter modules. This analog architecture has many similarities to the idealized system shown in Figure 1, though it omits the IF stage, and also the separate transmit amplitude modulation stage. Both amplitude and phase modulation are accomplished with the I & Q DACs. This design might appear to be a step backward from the all-digital rf systems that have become commonplace. However, some of the other features of the device mitigate the imperfections associated with the analog design. First, automated on-chip calibration of I & Q channel amplitude, offset, and phase errors minimizes quadrature artifacts in both the transmit and receive channels. More importantly, the transceiver signal processing (TSP) modules make the LMS7002M a hybrid analog-digital system that provides many of the advantages of a modern digital system. In particular, the TSP modules contain numerical oscillators and mixers that allow the LO frequency to be shifted several MHz away from the NMR frequency, so that any residual artifacts arising from amplitude and phase errors in the analog stages are shifted far outside of the spectral region of interest. For transmit, the TSP modules implement digital frequency shifting by manipulating the values sent to the DACs. For example, when a constant amplitude and phase rf output is desired, the TSPs will provide time-dependent numerical values to the DACs so that their outputs oscillate sinusoidally, in quadrature. This frequency shifting amounts to a digital version of the IF described for the idealized system of Figure 1. Instead of frequency shifting in a separate analog mixing stage, however, the frequency shifting here is performed by the same mixers that do the I & Q modulation, while the IF oscillation is introduced by the DACs.

On the receive side, incoming signals are mixed with the quadrature LO signals, then low-pass filtered and digitized, again in a fashion similar to the idealized system of Figure 1. After digitization, however, the TSP again implements a digital IF frequency shift by multiplying the ADC outputs with the in-phase and quadrature outputs of the numerical oscillator. This numerical frequency shift moves any artifacts due to imperfect analog quadrature detection by twice the IF frequency. The remaining artifacts can then be easily eliminated by straightforward digital filtering. It may be instructive to think of this frequency shift in comparison to the phase cycling that is commonly used with pure analog receivers. Because the inputs to the ADCs are

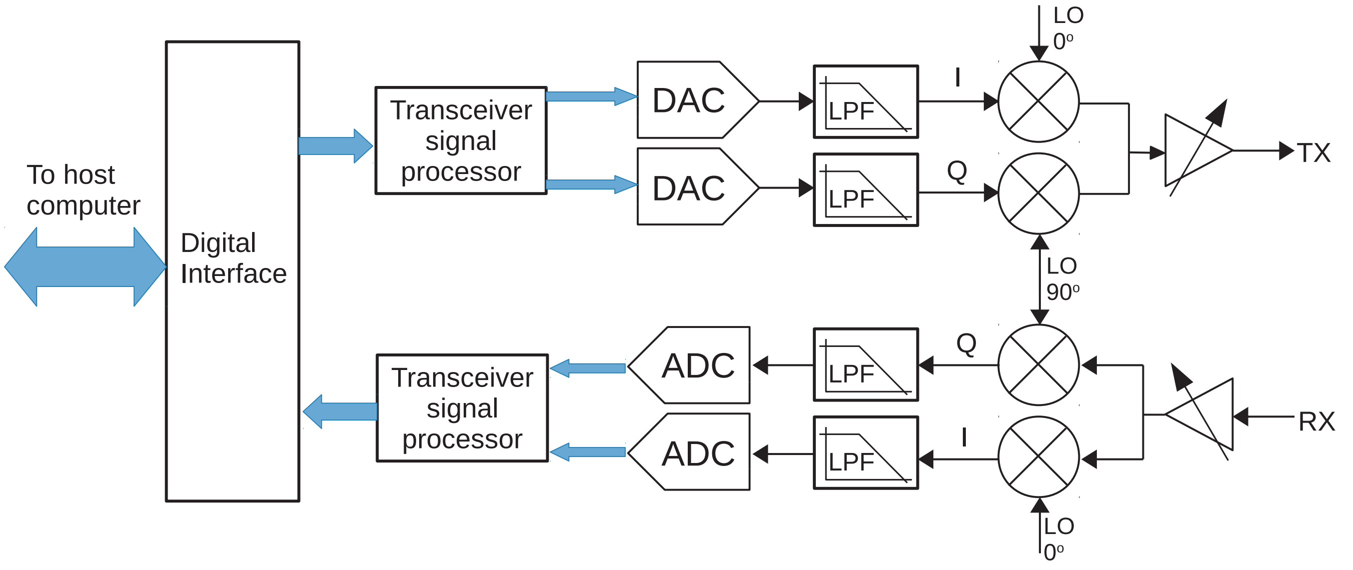


FIGURE 2 Highly simplified diagram of the internal architecture of the LMS7002M device. The chip contains two complete transmit and receive paths, only one of which is illustrated here. The present spectrometer implementation only makes use of a single transmit and receive channel from each device. There are restrictions on how far apart the frequencies of the two channels can be, described in the text

in quadrature at a high frequency (compared to the range of NMR frequencies), the incoming NMR signals rapidly oscillate between the two ADC channels. This oscillation averages any imbalances between the channels, and is conceptually similar to phase cycling, except that it happens within a single transient.

In addition to digital frequency shifting, the TSP modules can provide decimation (receiver channels) or interpolation (transmit channels) to allow the internal ADCs and DACs to run at higher conversion rates than the sample rate provided to/from the host computer. They provide multiple stages of digital filtering, play a role in correction of the analog amplitude and phase imbalances, and provide several features more relevant for wireless communications such as dynamic automatic gain control and received signal strength measurements.

Digital data to specify the transmitter output are streamed to the device through the USB interface as a continuous flow of 12-bit I & Q values which, after processing by the TSP modules, are presented to the DACs at a rate specified by a sampling clock. Similarly, received data from the ADCs are streamed back to the host computer over the same USB interface. We have chosen to use sample data clocks of 10, 25, or 50 MSPS, in order to have all event timing aligned straightforwardly with the digital pulse programmer described below. At 25 MSPS, a single USB3 interface can support two transmit and receive streams running continuously. At 50 MSPS, a second USB3 interface will be required for all four streams to run continuously. However, a pulse sequence that has little transmit activity could be accommodated on a single USB3 bus.

The detailed hardware specifications of the LMS7002M device are provided in its data sheet[16] and will not be repeated here.

2.3 | Spectrometer architecture

To use the LimeSDR boards in a fully functional NMR console, several additional components are added. The present two-channel prototype spectrometer console consists of two LimeSDR boards, four rf gates, and an Arduino Due microcontroller board acting as a digital logic pulse programmer. A block diagram is shown in Figure 3. The system is easily expanded to incorporate additional rf channels by adding additional LimeSDR boards and rf switches. At present, a PTS 310 synthesizer (Programmed Test Sources, Littleton, MA) is used to provide a frequency reference for the SDRs and the Arduino. The PTS was used as it was immediately available, a simpler frequency reference source would work just as well. The Due requires a 12 MHz reference; the LimeSDR boards have a flexible reference clocking input and could use the same 12 MHz reference as the Due in place of the 10 MHz reference used here. The SDR boards are interfaced to a host computer over a USB3 interface while the Due has a native USB2 interface.

The inputs and outputs of the LimeSDR boards are connected to external pre- and power amplifiers through rf switches. On the receive side, the switch protects the LimeSDR input from the saturated preamplifier output during rf pulses, while the switches on the transmit side are not strictly necessary, but are included as a precaution to

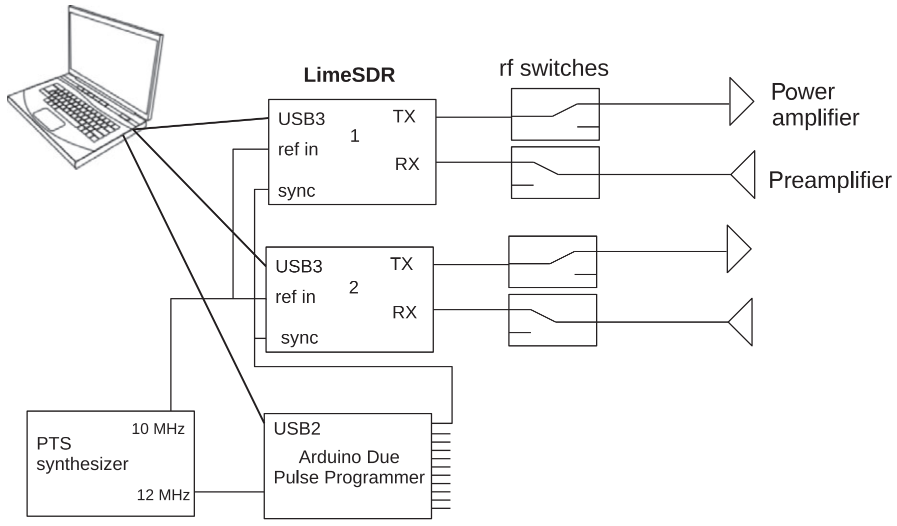


FIGURE 3 Spectrometer console block diagram. Two LimeSDR boards are used to accommodate the two independent rf channels, though if their frequencies were close enough, a single LimeSDR board could be used. The rf switches are Minicircuits ZYSWA-2-50DR. Connections for logic control of the rf switches and the power amplifier blanking gates are not shown. Additional rf channels can be installed by adding more LimeSDRs in parallel. The synchronization scheme is highly simplified here, details are shown in the Appendix

protect the NMR probe from any unexpected output from the SDRs.

The high level of integration of the LimeSDR boards means that a relatively small number of components is required to assemble a fully functional rf console. The pulse programmer, switches, and LimeSDR boards all fit easily into one compact 19” rack-mount aluminum enclosure. The total cost of the SDR components here, for two independent rf channels, is US$600, while Arduino Due boards can be found from numerous online vendors for ~$12. Notably, each LimeSDR board incorporates two complete transmit and two complete receive channels. Unfortunately, the rf device on the LimeSDR board incorporates only two rf synthesizers, and the internal organization of the device requires that the LO frequency of both transmitters be identical and that the LO frequencies of both receivers also be the same. All four rf channels can be drawn from a single internal LO source to ensure coherence between them. However, this does not mean that the two separate transmit and receive paths must be used at identical frequencies. The TSP numerical oscillators may be used to shift the frequency of any individual channel, allowing different (digital) IF frequencies for the different channels. The result of this is that a single board can be used for two NMR channels with a frequency difference up to ~60 MHz. Our implementation at this point only makes use of a single transmit and receive channel on each board, but at lower magnetic field strengths, such as found in recent generations of desktop permanent-magnet NMR systems, a single board could supply the entire rf transceiver system for common ^{1}H-^{13}C double-resonance experiments. For imaging systems with multiple receive coils operating at the same frequency, each LimeSDR board could act as receiver for two coils. As multi-transmit is becoming more common at very high-field in MRI,[17] each LimeSDR could also act as a transmitter for two coils.

2.4 | Arduino Due-based pulse programmer

In addition to rf transmit and receive capabilities, a general-purpose NMR spectrometer requires digital logic outputs to control the blanking of high-power amplifiers, switch receiver input gates, and control any other devices required for a particular experiment. The LimeSDR board does include eight GPIO pins which in principle could be used for these purposes, however the software stack supporting the LimeSDR board is not organized for these pins to be controlled with sufficient timing precision to be useful. Because the entire software stack is open-source, modifications could be made to address this; however, extremely invasive changes would be required to several different layers of the software, and would likely be challenging to maintain as the software stack evolves in the future.

Instead, inspired by a recent paper that employed an Arduino Due microcontroller as a digital pulse sequence generator,[18] we have constructed an inexpensive, versatile pulse programmer using the same microcontroller platform. The device described in reference 18 makes use of assembly-language delay loops to provide precisely timed intervals for a set of digital outputs. That implementation optimizes both the minimum-event duration (24 ns) and the number of events in a sequence (~20 000). The design used there however was deemed to be inappropriate for a general-purpose NMR spectrometer because each sequence must be flashed into the microcontroller's flash memory. The time taken for this process is on the order of a minute for a lengthy program. In addition to this unacceptable time constraint, the flash memory used in these devices has a limited number of write cycles, which would be rapidly approached if new sequences were frequently flashed in place.

A somewhat different approach was employed here. Firmware for the Due was written that executes a pulse program that is stored in RAM rather than in flash. Pulse programs are then downloaded over the native USB2 interface at high speed. A maximal length pulse program, consisting of about 12 000 events, can be downloaded in ~25 ms. This architecture allows for a great deal of flexibility in the use of the Due and in the instructions that may be included in the pulse program language. The pulse-program engine in the firmware supports unlimited levels of subroutines and nested looping (limited in practice by the RAM available to the stack for storing return addresses). The chosen configuration supports 25 digital outputs that may be set in events with a 200 ns minimum-event time and 20 ns event resolution. This implementation also supports 24 additional digital outputs and two 12-bit unipolar analog outputs that can be set either from within a running pulse program, with a longer minimum-event duration, or can be set asynchronously while the programmer is otherwise idle. There are four pins set aside as analog inputs which can be read by a 12-bit analog ADC while the programmer is idle. Many of these pins are configurable and could be easily reassigned if necessary. If additional I/O pins are necessary, multiple boards can be easily paralleled and synchronized. Another key feature is "non-stop" operation, which allows a sequence to be downloaded into RAM *during* the final event of the previous sequence, so that successive sequences begin with no delay.

To achieve the 20 ns event resolution, an internal timer peripheral is clocked at 50 MHz, requiring a 100 MHz CPU clock. This is a modest "overclocking" of the Due's microcontroller, but in fact the architecture of the software shuts the CPU clock off for the vast majority of time, with the CPU active only in brief intervals at the start and end of each event. An additional output pin of the Due is used to generate a clock signal for synchronization of the ADC/DAC clocks in the LimeSDRs, as will be described further below. The device also supports a clean abort facility, so that an executing pulse sequence can be terminated prematurely simply by sending an instruction over the USB interface.

For optimal reproducibility of the event timing, the Due outputs are latched with external 74AC574 latch chips, as shown in Figure 4, so that a single timer-controlled latch pin releases a new event word to the driven devices. Without the latches, there is a 10 ns jitter on the output event timing. With the latches, event timing reproducibility is limited by the clock source supplying the Due. For devices such as power amplifier blanking signals that are commonly driven over lengthy 50 Ω coaxial cables, TC4427 mosfet switches are used as line drivers, with 33 Ω resistors in series to protect from short circuits. Two latch chips serving 16 outputs along with five TC4427 drivers for 10 outputs are easily accommodated on an inexpensive prototyping "shield" that seats directly onto the Arduino Due. Including the prototyping board, the latch, and driver chips and the Due, the pulse programmer hardware total cost is about $20 plus cabling and connectors.

Two external trigger inputs are provided; one can be used for beginning pulse program execution, allowing multiple boards to be started synchronously, the other is for in-sequence synchronization with external events and can be instructed to wait for a chosen maximum interval, or to wait indefinitely.

2.5 | Synchronization

For reproducible NMR experiments, rf and logic signals from multiple LimeSDR boards and the Due pulse programmer must be accurately synchronized. This issue has not been commonly addressed in publications of previous SDR-based NMR systems, and where it has, it appears to have been a serious problem, for example in Ref. 6, where the detection of and recovery from "de-synchronization events" are described. A great deal of effort has gone into ensuring accurate and consistent timing of the outputs of the various devices used here. The ability to precisely and reproducibly synchronize multiple rf channels with each other and with logic signals may be the most significant aspect of this work.

A prerequisite for useful synchronization is a common clock source. Supplying the LimeSDR boards and the Arduino Due with reference clocks drawn from the same stable oscillator satisfies this requirement. In order to clock the Due from an external reference, a minor modification to its circuit board was required. Its on-board 12 MHz oscillator and associated load capacitors were removed, then a small diameter coaxial cable and 50 Ω termination resistor were soldered on in their place.

Time on the Due is incremented in 20 ns timer ticks, where the 20 ns interval is the period of the 50 MHz timer clock. On the LimeSDR board, the timing granularity of the rf modulation is set by the ADC/DAC sample clocks. Sample frames are counted by a timestamp counter within the FPGA. Timestamps are then reported for each received sample, and timestamps may be specified for transmitted samples.

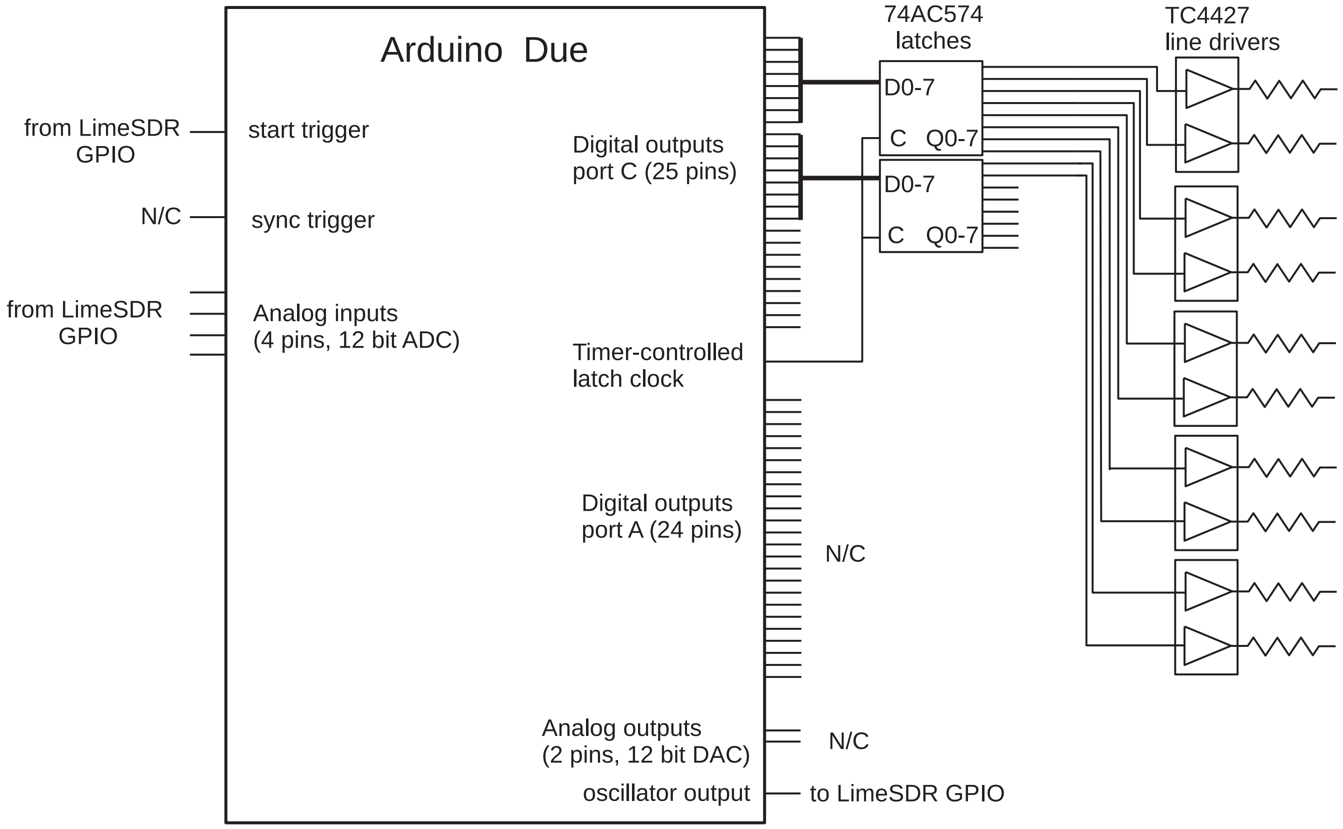


FIGURE 4 Arduino Due Pulse Programmer hardware. The latches and line drivers shown are easily accommodated on a small prototyping “shield” that mounts directly on top of the Arduino Due

The common reference clock ensures that time proceeds at the same rate on each device, however it does not guarantee any particular relationship between event timings on the various devices. In order to ensure predictable, reproducible event start times, modifications to the LimeSDR's FPGA firmware were required. The modifications are described in detail in the Appendix. Briefly, submultiples of the ADC/DAC sample clocks are compared to an output of the Arduino Due, configured to run at the same frequency, using an XOR-gate phase detector. Manipulation of the ADC/DAC frequency allows the various clocks to be locked together in frequency and phase to provide nanosecond reproducibility in the timing of events occurring on the different devices.

3 | SOFTWARE

The software to run the spectrometer consists of a number of different components. The various layers and their connectivities are shown in Figure 5.

3.1 | Software: Arduino Due

The Arduino Due runs custom firmware written in a mix of C/C++, inline assembly, and preassembled binary code. The entire source code for the Due is a single file, managed as a “sketch” within the open-source, multi-platform Arduino integrated development environment.[19] Compiling the binary pulse programs for the Due programmer and controlling its execution are facilitated by a small C-language library. Both the firmware and support library are licensed under the GNU general public license, and are freely downloadable from http://www.phas.ubc.ca/~michal/DuePulseProgrammer.

3.2 | Software: LimeSDR

The LimeSDR board itself has four distinct software/firmware components. The FX3 USB3 interface chip has field replaceable firmware, as does the Cyclone IV-E FPGA. Within the FPGA, a NIOS II microcontroller is emulated; its code takes part in configuring components within the FPGA, some of the other peripherals on the LimeSDR board, and programming registers within the LMS7002M. Finally, the LMS7002M chip has within it an 8051-based microcontroller that runs code involved in on-chip calibration. On the host computer, the LimeSuite software provides a C/C++ library for controlling the LimeSDR devices, along with a graphical user interface that can be used to configure and test the LimeSDR boards. LimeSuite

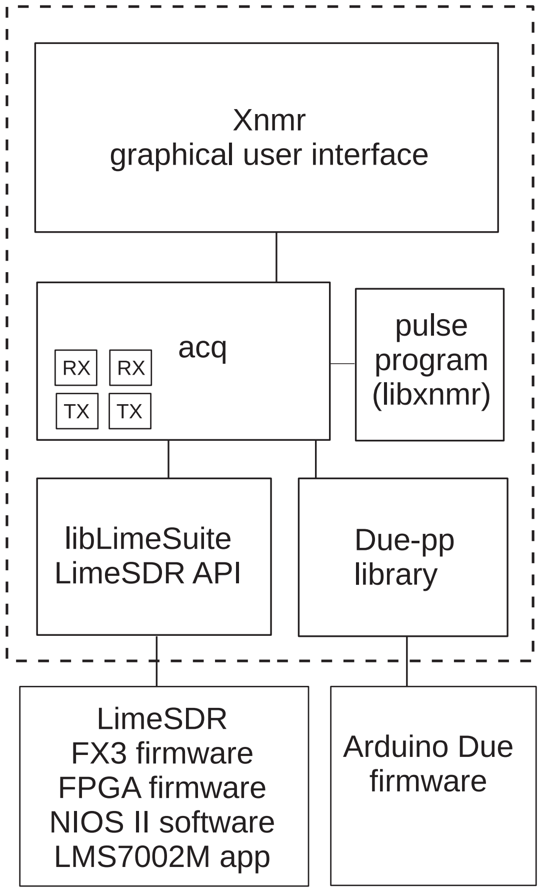


FIGURE 5 Software architecture. Elements inside the dashed box run on the host computer, while those below execute on systems within the LimeSDR or the Arduino Due

also supplies a utility that can be used to flash new firmware into the USB3 interface and the FPGA.

All of this code associated with the LimeSDR is open-source, released by its authors under the Apache License, and may be modified and redistributed. The modifications to the LimeSuite library and the FPGA firmware that were required for synchronization of the ADC/DAC sample clocks with the Due are also available for download, and can be found at http://www.phas.ubc.ca/~michal/limeSDR.

3.3 | Software: Xnmr

Above the hardware interface libraries, our home-built NMR acquisition software runs as three distinct programs. There is a GTK+[20] based graphical user interface that allows the user to choose a pulse program, set acquisition parameters, control the spectrometer, and perform basic NMR processing tasks. Below the user interface, a separate acquisition program, *acq*, receives instructions from the user interface, handles all interactions with the hardware via the hardware abstraction libraries, and manages the pulse program itself. Separate threads are launched to handle transmit and receive tasks on each LimeSDR board. Pulse programs are written as stand-alone C programs that interact with *acq* through a small library. "Real-time" scheduling is used under a standard desktop linux kernel to prioritize the threads that communicate with hardware. While this scheduling feature does not provide strict real-time scheduling, it is sufficient to allow a desktop computer to run the NMR spectrometer and other common desktop programs simultaneously without issue.

3.4 | Pulse programming language

The pulse programming language consists of a small number of functions to allow the pulse program to communicate with *acq*, to control the output of the Due's digital logic outputs, to control the rf outputs of the LimeSDRs, and to control the timing and accumulation of received data. The software includes rudimentary support for acquiring data from multiple receivers. The language supports the hardware looping and subroutine features of the Due pulse programmer, and treats the separate pieces of hardware in a unified fashion. The body of a simple one-pulse pulse program is shown in Figure 6. Any C-language features can be used, so complicated phase cycles can be generated "on-the-fly" and involved mathematical calculations can be undertaken within the pulse program. To perform a hardware event, a line of code begins with the `EVENT` identifier, followed by the duration of the event. Following the duration, a series of zero or more hardware devices may be addressed. Each device is referenced with a `{DEVICE, PARAMETERS}` construct. For logic devices there is always a single numerical parameter. For transmitters there are two parameters, an amplitude and phase, while for receive instructions only a phase is necessary. A preprocessor turns these EVENT lines into function calls that do the work required to control the specified hardware.

The pulse program executes one iteration through its main loop for each transient, ahead of actual sequence execution. The EVENT lines in the program populate tables in the host computer memory. There is one table for each LimeSDR containing all the transmit and receive events for that board, and a separate table for the events to be executed on the Due. When the pulse program has completed its iteration, the Due program is compiled and downloaded, and the LimeSDR event tables are transferred to the transmit threads. The transmit threads generate DAC samples in real time while the receive threads receive steady streams of data from the LimeSDRs ADCs. Most of these streams are discarded, only data requested by receive events in the pulse program are retained. Subsequent iterations by the pulse program take place while the previous iteration's table is executing on the hardware. This architecture allows subsequent transients to begin precisely at the end of the

```
    int ph_rec[4]={0,2,1,3};            // receiver phases
    float  ph_tr[4]={0.,180.,90.,270.}; // pulse phases
    pulse_program_init();
    do {
     // fetch parameters from gui:
      GET_PARAMETER_FLOAT( sf1offset );
      GET_PARAMETER_FLOAT( amplitude1 );
      GET_PARAMETER_FLOAT( pw1 );
      GET_PARAMETER_FLOAT( final );
      GET_PARAMETER_FLOAT( rd );
      GET_PARAMETER_FLOAT( d1 );
      GET_PARAMETER_DOUBLE( sf1 );
      GET_PARAMETER_INT( rgain );
      GET_PARAMETER_INT( tgain );
      acqn = get_acqn();

      begin();
      tx_init(0,sf1*1e6+sf1offset,2.4e6,tgain);// board #, freq, nco f, gain.
      rx_init(0,1./get_dwell(),rgain);         // board #, sw, gain
      EVENT d1;
      EVENT 2e-6 {GATE1,1} {BLNK1,1};
      EVENT pw1  {TX1,amplitude1,ph_tr[acqn%4] } {GATE1,1} {BLNK1,1} ;
      EVENT rd;
      EVENT get_npts()*get_dwell() {RX1,90*ph_rec[acqn%4]} {RCVR_GATE1,1};
      EVENT final;
      } while( ready(0) == P_PROGRAM_CALC );
  done();
  }
```

FIGURE 6 Complete pulse program example for a simple one-pulse NMR experiment

currently running sequence, with no unexpected or variable delays. While the Due can only store programs having a maximum of 12 000 events, its in-built looping and subroutine capabilities, along with a feature of its compiler that compresses identical events, mean that in general, sequence length and complexity are limited by the resources of the host computer. This is largely due to the fact that amplitude and phase changes for modulated pulses do not incur additional events on the Due, as they are managed completely on the SDR boards. Sequences having millions of elements are easily managed with a reasonably modern host computer.

4 | EXPERIMENTAL METHODS

NMR data were collected using a 4.7 T Oxford Instruments 89 mm bore superconducting magnet equipped with a 12 gradient shim set. Solid-state NMR spectra were collected using a Bruker HP WB73A SOL10 probe, while a WP200 probe was used for solution spectra. Unless otherwise noted, received signals were first amplified by the preamplifier from a Bruker MSL-200 console, and pulses to the probe were amplified with an AMT 3900 power amplifier from a decommissioned Varian console.

The LimeSDR boards and Arduino Due pulse programmer were all connected to USB ports on ~3-year-old Intel Core i5-4690 quad core computer running Gentoo Linux. This computer ran the user interface as well as all of the real-time communications with the spectrometer components.

5 | RESULTS

Some examples of the transmitter output from the LimeSDR boards are shown in Figure 7. Shaped pulses are shown in Figure 7A, where 3.2 and 32 μs long Gaussian pulses at 200.6 MHz were generated using 200 ns long steps. In the former, the steps in the rf amplitude are

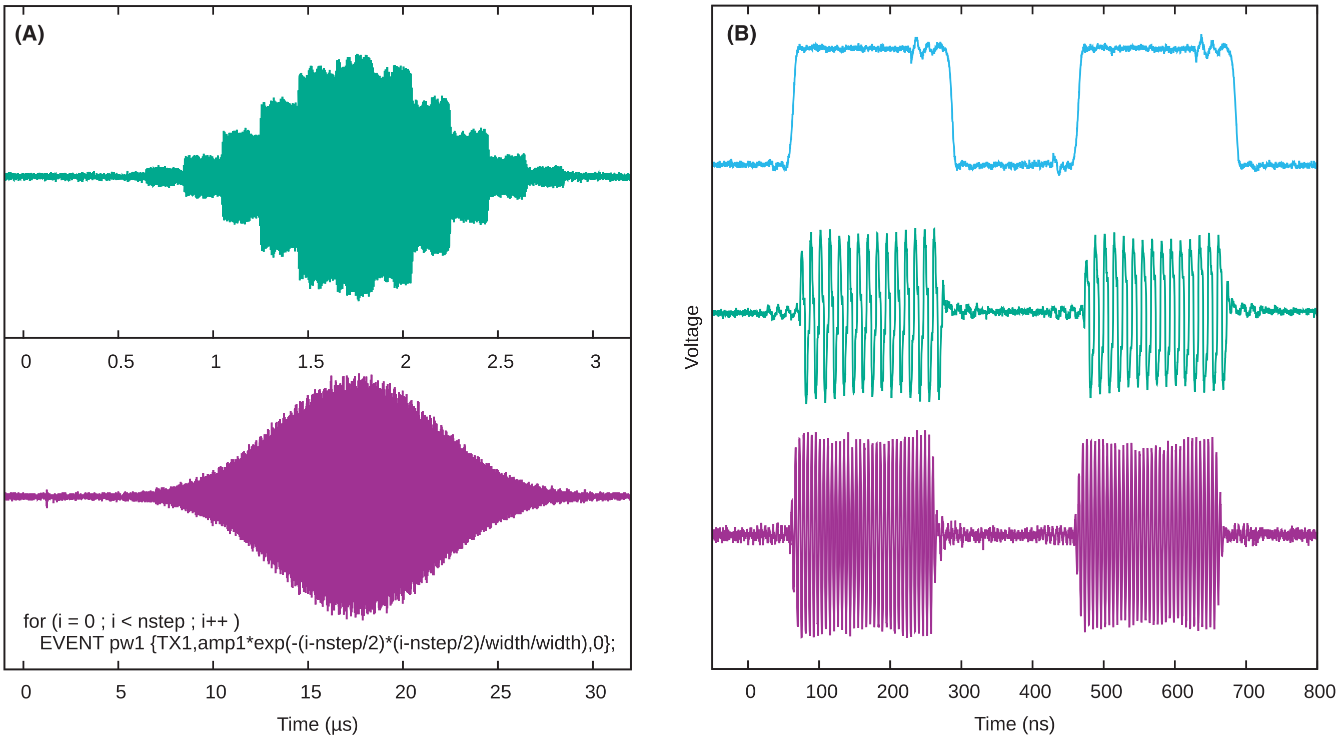


FIGURE 7 Example output of transmitters. In (A), Gaussian shaped pulses were produced using the two lines of pulse programming code shown. In the upper trace, `width` was set to 3 and `nstep` to 16. In the lower trace, both values were 10 times larger, and the *x*-axis has been compressed by a factor of 10. In (B), 200 ns long pulses from the Due and two LimeSDRs captured simultaneously demonstrate the synchronization of the outputs of the different devices. The data were captured with a Tektronix TDS3054B oscilloscope

clearly visible, while in the latter, the rf amplitude appears to change very smoothly. If smoother steps are desired for very short pulses, the time step duration can be decreased to 20 ns, allowing extremely smooth changes in the rf amplitude and phase. The pulse programming code required to produce the pulses is shown in the figure.

In Figure 7B, the synchronization of two different LimeSDR boards, with output frequencies of 176 and 75 MHz, along with a logic output from the Due is demonstrated. Two hundred nanosecond long pulses from each device, captured simultaneously, are shown. The start times are reproducible to better than 1 ns.

A high-resolution NMR spectrum of a chloroform in acetone-d_6 sample, apparently containing a small ethanol impurity, is shown in Figure 8. An inset shows the region of the spectrum where an image of the main chloroform peak would be expected from a traditional analog quadrature receiver. The main peak here has a signal-to-noise ratio of 2250, but no discernible image peak is observed. The numerical oscillators in the TSPs were used to shift the LO frequency 2.4 MHz away from the desired center of the spectrum, so any quadrature image peaks are shifted by 4.8 MHz from the center of the spectrum.

A wide-line spectrum was obtained from a $CaSO_4 \cdot \frac{1}{2}H_2O$ (Sigma-Aldrich, St. Louis, MO, USA) sample using single

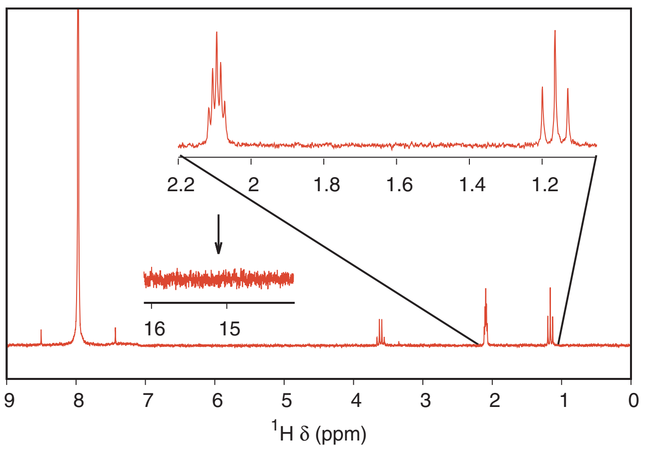
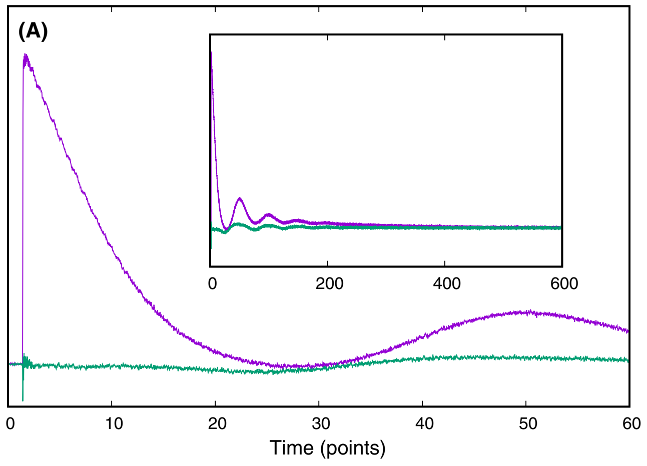


FIGURE 8 Single-transient high-resolution 200 MHz ^{1}H NMR spectrum of a 20% chloroform in acetone-d_6 sample (Varian part number 968120-76). The vertical scale has been expanded to show the small acetone and impurity ethanol peaks so that the main chloroform peak has been truncated. The upper inset shows an expansion to better reveal two of the multiplets. The transmitter frequency was set to 11.54 ppm in this spectrum. The vertical arrow above the lower inset shows the location where an image of the main chloroform peak would be expected from a traditional analog quadrature receiver. With the digital frequency shift of 2.4 MHz, any quadrature image peaks would be shifted by 4.8 MHz from the spectrum



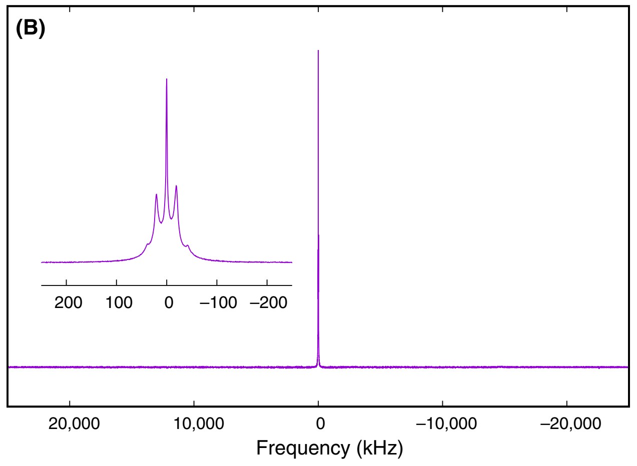


FIGURE 9 Wide-line 200 MHz ^{1}H NMR data of $CaSO_4 \cdot \frac{1}{2}H_2O$. In (A), the free induction decay, acquired with a dwell of 20 ns per point is shown. The main figure shows the initial 60 μs of the data (3000 points at 50 MSPS), the inset shows the entire 600 μs, 30 000 point data set. In (B), the 50 MHz wide spectrum, corresponding to the fid in (A) is shown. The inset shows an expanded view of the center 500 kHz, showing the expected Pake pattern. These data are the result of co-adding 2000 transients

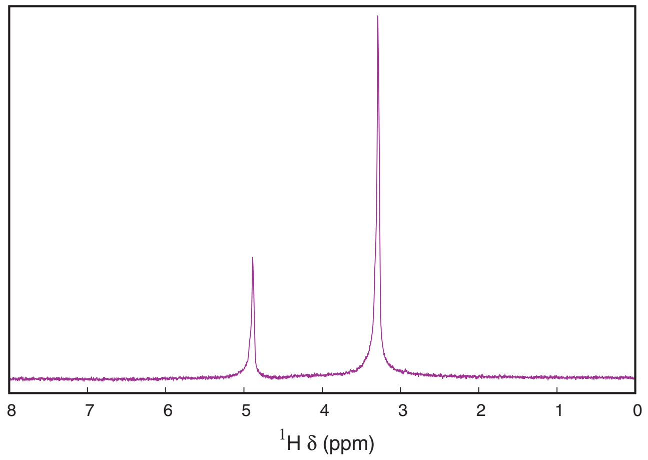


FIGURE 10 Single-transient ^{1}H NMR spectrum of 100% methanol, acquired without external power amplifier or preamplifier, using only components built into the LimeSDR board

pulse acquisition, and is shown in Figure 9. Here, the spectral width was set to 50 MHz. The rise at the start of the acquisition (Figure 9A) when the receive gate is switched open is very clean due to the absence of narrow-band filters in either the digital or analog domain. The ~100 kHz wide Pake pattern can be observed in the inset of the spectrum in Figure 9B. When the whole spectral width is shown, the 100 kHz wide powder pattern appears as a sharp peak.

Finally, to demonstrate the high level of integration of the LimeSDR board, a spectrum of a methanol temperature calibration sample (Varian part number 968120-80) was collected without external power or preamplifiers, and is shown in Figure 10. The transmit and receive ports of a LimeSDR board were connected to a Minicircuits ZYSWA-2-50DR rf switch, the common port of which was connected directly to the NMR probe. The Due pulse programmer was used to switch between transmit and receive. The switch could be replaced with a passive device such as a rf circulator, eliminating the need for the Due pulse programmer to make a minimal spectrometer consisting of only the LimeSDR board and the circulator. Omitting the preamplifier does have an impact on the signal-to-noise ratio achievable, as the noise figure specification of the LimeSDR's low noise amplifier, 2.0 dB, is not as low as those found typically on NMR preamplifiers. Omitting the power amplifier has a significant impact on the achievable strength of the rf pulses. Such an arrangement might be useful for very small or concentrated samples.

6 | DISCUSSION

In comparison to NMR systems built from earlier software-defined radio devices, the present system has two important advantages. First, the LMS7002M chip incorporates an astounding level of integration. The one chip is essentially two complete transmitters and receivers. Second, the open-source nature of both the software and the support firmware allows modification of the device's capabilities to optimize it for NMR-specific applications. This has allowed for the precise synchronization of multiple rf and digital logic devices. The spectrometer described here could replace the rf console of a commercial multi-channel high-field NMR system. Previously reported SDR-based NMR systems incorporated only a single rf channel. The broadband system described here is capable of multi-nuclear solution and solids NMR experiments at any but the very lowest magnetic field strengths.

Compared to a fully digital rf system, the hybrid analog/digital design of the LMS7002M offers a fully broadband device, able to operate from near DC to 3.8 GHz, but does have the potential to introduce artifacts due to offsets and imbalances between quadrature channels. These artifacts however are easy to manage, thanks to the numerical frequency shifting of the TSP which moves any such artifacts out of the spectral window. Another limitation of the present system arises from the internal architecture of the LMS7002M, where a pair of 12-bit I & Q DACs is used for both phase and amplitude modulation. The result of this is that phase modulation precision degrades as the amplitude of the rf output is reduced.

The Arduino Due-based pulse programmer presents a substantial advance in the price/performance equation for a digital logic programmer. There is one commercial product it is easily compared to, the PulseBlaster™ (SpinCore Technologies, Gainesville, FL). The PulseBlaster offers 24 digital outputs, a 4096 event memory, 50 ns minimum pulse length, and a pulse resolution of 10 ns. Compared to the Due-based pulse programmer, the PulseBlaster offers a minimum-event length that is 1/4 the length, and a timing resolution that is better by a factor of 2. However, the maximum number of events in a program is about 1/3 that of the Due. The capabilities for looping and subroutines are comparable, though slightly more flexible on the Due, as the PulseBlaster limits to eight nested levels. The PulseBlaster lacks several features incorporated in the Due, most significant of which is "non-stop" operation, allowing a successive experiment to be downloaded during the final event of the previous sequence. The Due also offers two analog outputs, four analog inputs, and an additional 24 digital outputs. Both are cross platform and can be expanded by paralleling additional boards. The Due has a more flexible scheme for synchronizing to external events. Perhaps the two most significant differences are the shorter minimum-event length and time resolution of the PulseBlaster (a factor of 2 or 4 in its favor), and the price difference, a factor of ~130 (US $1525 vs $12) in the Due's favor. In addition, the PulseBlaster is offered by a single company, while Arduino Due boards are manufactured by, and available from, numerous suppliers in different parts of the world.

Other microcontroller products comparable to the Arduino Due exist that would likely offer even better performance, with only slightly increased cost. As an example, the Teensy 3.6 (US$30, PJRC[21]), offers significantly faster clock speeds and more RAM (ie, longer programs) than the Due, and would likely make a very good pulse programmer. The choice was made here to use the Due for several reasons: familiarity with device, multi-supplier availability, the fully open-source nature of the development platform, and the fact that its specifications are adequate for our needs.

Since this project was begun, an even less expensive version of the LimeSDR, called LimeSDR Mini, that is based upon the same rf chip and host driver library, has been released. The host computer software developed here should require only minimal modifications to make use of this newer board. However, the FPGA on this board is different, and its firmware has not yet been modified to enable the precise synchronization available with the LimeSDR.

7 | CONCLUSIONS

We have constructed a flexible, expandable, multi-channel transmit/receive console for NMR based upon an inexpensive, open-source software-defined radio platform and Arduino Due-based pulse programmer. This is believed to be the first report of a multi-channel NMR spectrometer based on a commercial SDR. The system provides a powerful but simple to use pulse programming language and is able to execute pulse sequences of nearly arbitrary length and complexity. Despite the inherent disadvantages of analog vector modulation and quadrature detection compared to contemporary digital rf designs, the system provides competitive performance due to its digital frequency shifting capabilities. The open-source nature of the device firmware and software proved to be essential in order to provide precise synchronization of multiple devices for multi-channel operation using a hobbiest-level SDR platform. The Arduino Due-based pulse-programmer provides remarkable performance for relevant NMR needs at very low cost.

ACKNOWLEDGMENTS

This work was supported by a Discovery Grant from the Natural Sciences and Engineering Research Council of Canada. We thank Victor Soutyrine for modifying the Arduino Due, Radim Hošák and Miroslav Ježek for code samples from their device and advice on supplying external clock signals to the Due, and Stefan Reinsberg for a review of the manuscript.

ORCID

Carl A. Michal https://orcid.org/0000-0002-5764-3221

REFERENCES

1. Hibino Y, Sugahara K, Muro Y, Tanaka H, Sato T, Kondo Y. Simple and low-cost tabletop NMR system for chemical-shift-resolution spectra measurements. *J Magn Reson*. 2018;294:128-132.
2. Yu P, Xu YJ, Wu ZY, Chang Y, Chen QY, Yang XD. A low-cost home-build NMR using Halbach magnet. *J Magn Reson*. 2018;294:162-168.

3. Biller JR, Stupic K, Kos AB, et al. Characterization of a PXIe based low-field digital NMR spectrometer. *Concepts Magn Reson B*. 2017;47B:e21350.
4. Chen HY, Kim Y, Nath P, Hilty C. An ultra-low cost NMR device with arbitrary pulse programming. *J Magn Reson*. 2015;255:100-105.
5. Michal CA. A low-cost spectrometer for NMR measurements in the Earth's magnetic field. *Meas Sci Technol*. 2010;21:105902.
6. Hasselwander CJ, Cao Z, Grissom WA. gr-MRI: a software package for magnetic resonance imaging using software defined radios. *J Magn Reson*. 2016;270:47-55.
7. Tang W, Wang W. A single-board NMR spectrometer based on a software defined radio architecture. *Meas Sci Technol*. 2011;22:015902.
8. Asfour A, Raoof K, Yonnet JP. Software defined radio (SDR) and direct digital synthesizer (DDS) for NMR/MRI instruments at low-field. *Sensors*. 2013;13:16245-16262.
9. Tian M, Liu H, Chen Z. *Application of software defined radio in 500 MHz NMR spectrometers. The Ninth International Conference on Electron Measurement & Instruments*. 2009;1-800-1-804.
10. Ha D, Paulsen J, Sun N, Song YQ, Ham D. Scalable NMR spectroscopy with semiconductor chips. *Proc Natl Acad Sci USA*. 2014;111:11955-11960.
11. Ha D, Sun N, Paulsen J, et al. *Integrated CMOS spectrometer for multidimensional NMR spectroscopy. In: 2017 IEEE 60th International Midwest Symposium on Circuits and Systems*, 1085-1088.
12. Andres J, Chiaramonte G, SanGiorgio P, Boero G. A single-chip array of NMR receivers. *J Magn Reson*. 2011;209:1-7.
13. Kim J, Hammer B, Harjani R. *A 5-300 MHz CMOS transceiver for multi-nuclear NMR spectroscopy. In: Proc. IEEE 2012 Custom Integrated Circuits Conference*, https://doi.org/10.1109/cicc.2012.6330645.
14. *LimeSDR Crowd Supply Homepage*. https://www.crowdsupply.com/lime-micro/limesdr. Accessed June 4, 2018.
15. Reichert D, Hempel G. Receiver imperfections and CYCLOPS: an alternative description. *Concepts Magn Reson B* 2002;14:130-139.
16. *LMS7002M FPRF MIMO Transceiver IC with integrated microcontroller Datasheet, v3.1r00*. Lime Microsystems, Surrey, UK. 2017.
17. Brink W, Gulani V, Webb A. Clinical applications of dual-channel transmit MRI: a review. *J Magn Reson Imaging* 2015;25:855-869.
18. Hošák R, Ježek M. Arbitrary digital pulse sequence generator with delay-loop timing. *Rev Sci Instrum* 2018;89:045103.
19. *The Arduino homepage*. https://www.arduino.cc. Accessed June 12, 2018.
20. *The GTK+ Project*. https://www.gtk.org. Accessed June 12, 2018.
21. *PJRC Electronic Projects*. https://www.pjrc.com. Accessed June 12, 2018.

How to cite this article: Michal CA. A low-cost multi-channel software-defined radio-based NMR spectrometer and ultra-affordable digital pulse programmer. *Concepts Magn Reson Part B*. 2018; e21401. https://doi.org/10.1002/cmr.b.21401

APPENDIX

Synchronization

A number of modifications to the code stack supporting the LimeSDR board were required in order to obtain acceptable synchronization between multiple boards and with the Due.

In addition to a common time-base supplied by the reference clock, a common zero of time must be established amongst all relevant devices. The Due firmware was designed to accommodate such synchronization. The LimeSDR, however, was not designed with synchronization to external devices in mind.

The approach taken was to use a trigger on one of the LimeSDR's GPIO pins to indicate an internal timestamp "capture" event. While the GPIO pin is held high, all data packets of received samples relayed to the host computer carry the timestamp of the last sample from before the trigger. To indicate this capture to the host, the most significant bit (MSB) of the 64-bit timestamp is set high. The receiving software then only needs to recognize that the MSB is set and remove it to find the timestamp at which the trigger occurred. When the GPIO pin is brought low again, normal timestamps are restored.

This mechanism allows synchronization to within one sample period, but on its own is not sufficient. At a sample rate of 10 MSPS, 100 ns synchronization was not deemed to be acceptable. A scheme was developed to allow the phase of the sample clock in the LimeSDR to be locked to an external clock signal. A pulse width modulator module within the Arduino Due was used to generate a square wave signal at a submultiple of the sample clock frequency (eg, 10 MHz/16).

This signal was supplied to a GPIO pin on the LimeSDRs, and compared with divided-down representations of the sample clocks received from the LMS7002Ms in XOR gates acting as phase detectors. The outputs of the XORs are then low-pass filtered and connected to ADC inputs on the Due, as shown in Figure 11. In this fashion, the Due is able to measure the relative phases between its own PWM-generated sample clock and the LMS7002M sample clocks. To synchronize the clocks, small temporary adjustments are made to the frequency of the LMS7002M sample clock frequencies in order to align their phases with the Due.

This procedure allows the receive sample clocks to be aligned with the Due's clock to within 1 ns. Even after the receive sample timestamps are precisely synchronized, variations in the transmitter sample timing were observed. The synthesizer that drives the ADC and DAC sample clocks actually produces an output at a multiple (eight or more, depending on the decimation/interpolation factor desired) of

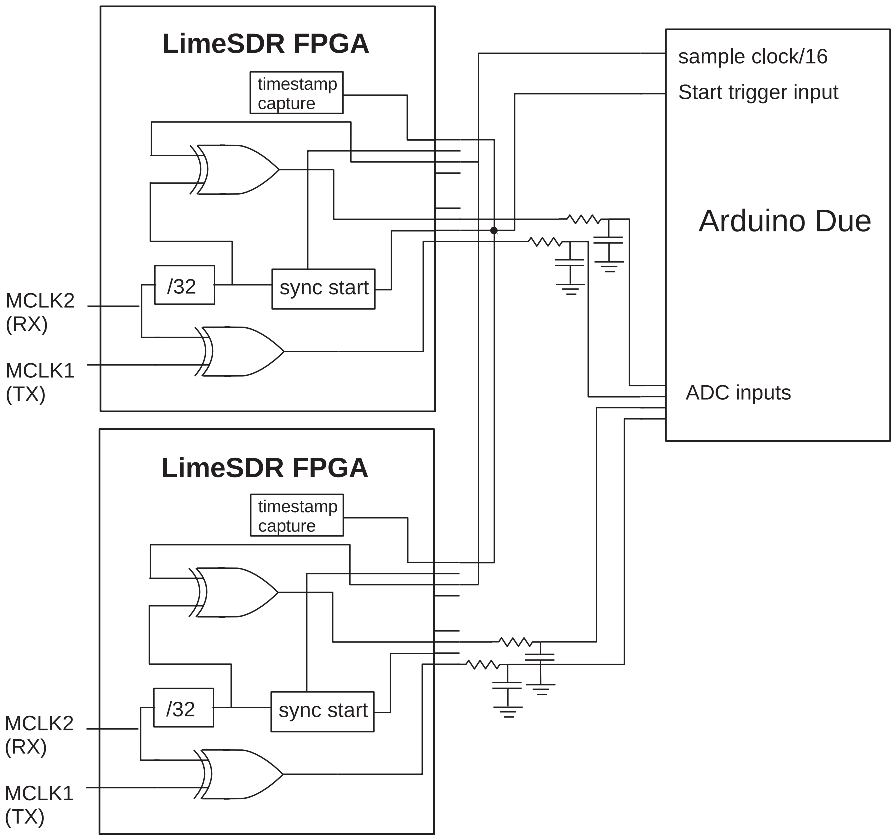


FIGURE A1 LimeSDR/Due Synchronization scheme. The LimeSDR's eight GPIO pins are shown on the right hand side of the symbol representing the LimeSDR's FPGA. The receive clocks on the LimeSDRs are divided by 32 because the sample clocks operate at twice the sample frequency to accommodate the transfer of four data words per sample frame. The MCLK signals are the receive and transmit sample clocks from the LMS7002Ms

the host interface sample rate. This clock is divided down by separate dividers on the receive and transmit channels, and during on-chip calibration operations these dividers can become out of step with each other. In order to synchronize the transmit samples, another XOR gate was implemented within each FPGA, and used to compare the receive sample clock to the transmit sample clock. The output of this XOR gate is provided on another of the GPIO pins, again low-pass filtered, and connected to an additional analog input on the Due. With this final modification to the FPGA, the transmit and receive sample clocks in multiple LimeSDR boards can be brought into precise, reproducible synchronization.