On Design and Testing of a Spectrometer Based on An FPGA Development Board for use with Optimal Control Theory and High-Q Resonators

by

Steven Casagrande

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Science
in
Physics

Waterloo, Ontario, Canada, 2014

Author's Declaration

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Abstract

Recent developments in quantum information processing have presented new and interesting ways to perform advanced algorithms and improve signal to noise ratios. Examples of these include optimal control theory pulse generation algorithms and the usage of high Q-factor resonators. However, these developments are blocked by current spectrometer designs. This thesis details the design and testing of a new spectrometer with sufficient accuracy, bandwidth, and control to implement these advances. The proposed solution is to use an FPGA-based development board together with custom computer software. This gives access to high-speed analogue inputs and outputs, as well as digital output pins. The spectrometer is then used in two X-band electron spin resonance experiments, showing how the advantages of the system allow for superior results to that possible with the previous equipment. In addition, the setup is used in a Nitrogen Vacancy (NV) system where a rabi experiment is performed.

Acknowledgements

I would like to thank my supervisor, Prof. David G. Cory, for accepting me into his freshly moved research group in the summer of 2010. His help and guidance over the past three years have been crucial in helping me understand the field and develop as a student.

I would also like to thank Prof. David Hawthorn and Prof. Jan Kycia for their role on my advisory committee.

The Institute for Quantum Computing USEQIP summer program selection committee of 2010 also deserve a thank you for if I had not been accepted into the program, I would not have had an opportunity to meet my (then future) supervisor.

I would like to particularly acknowledge those that helped with various aspects of this thesis: Chris Granade for all his help ranging from Windows development errors, hardware testing, and the development of key PC side software; Ian Hincks for his work with the PC side software; Troy Borneman for his help with performing experiments, his patience with software and hardware under development, and for putting up with sharing an office with me.

Finally, I would like to thank the love of my life, Carolyn Galvin, for supporting me all these years. Thank you Carolyn for being there for me during the trials and tribulations of graduate school.

Table of Contents

List of Tables vii

List of Figures viii

1 Motivation 1

1.1 Introduction . . . 2
1.2 Optimal Control Theory . . . 2
1.3 High-Q Resonators . . . 3
1.4 Current Spectrometers . . . 5

2 Solution 9

2.1 Introduction . . . 9
2.1.1 Field Programmable Gate Array . . . 9
2.2 Why Now? . . . 10
2.3 Innovative Integration X6-1000M . . . 11
2.4 Stock FPGA Configuration . . . 11
2.5 Custom FPGA Configuration . . . 13
2.5.1 Limitations . . . 16
2.6 DIO Breakout Board . . . 17
2.7 Python Communication Package . . . 20

3 Implementations **24**

3.1 Phase Calibration 24

3.2 ESR Spectrometer 25

3.2.1 Ringdown Compensation 26

3.2.2 Solid Echo 29

3.3 ODMR in Nitrogen-Vacancies 33

4 Conclusions **36**

5 Next Steps **37**

References **38**

APPENDICES **41**

A DIO Breakout Board Images **42**

B X6-1000M Images **46**

List of Tables

2.1 X6-1000M Specifications[16] 11

2.2 X6-1000M Analogue Input Specifications[16] 12

2.3 X6-1000M Analogue Output Specifications[16] 12

2.4 List of bit-widths for various parameters in the custom FPGA configuration by Innovative Integration. Delay is the time before pulse is enabled, width is how long it stays on for (see t_1 in figure 2.2), and a frame is equal to a signal "experiment". Therefore, frame length is the maximum time for one experiment to run, while frame count is the number of times a frame can be repeated. 16

2.5 List of miscellaneous figures related to the custom FPGA configuration by Innovative Integration. Resolution is key as it defines the minimum step one can adjust the delay and width parameters from table 2.4. 16

3.1 Results of spin-echo experiment. Measurements were taken with 8 averages each. Pulses were set to 60ns wide, with a 200MHz intermediate frequency. The static field applied to the cavity was 3360G. Pulses were not calibrated to be $\pi/2$. 32

List of Figures

1.1 Tank circuit block diagram. L and C_T are chosen such that their resonance is that of the desired pulsing frequency. C_M (the matching capacitor) is used to adjust the impedance of the circuit such that $Z_{in} = 50\Omega$. 3

1.2 Ringdown compensation using a basic negative phase pulse of short duration and high amplitude at the end. An infinite number of solutions exist for the compensation pulse width and amplitude. 5

1.3 Simplified diagram showing LO→RF leakage. Here the LO source is set to 10GHz at 10dBm, which is typical for X-band ESR applications. In this case the mixer has a LO→RF isolation of 50dB, causing a leakage of -40dBm at the RF port. 5

1.4 Simplified diagram showing circulator leakage. Here the leakage from the mixer (see figure 1.3) is input into the 40dB power amplifier. The circulator allows 20dB isolation leakage through, resulting in a -20dBm signal entering the receive circuitry. This leakage will be consistently present until the switch after the power amplifier is closed. 7

2.1 Diagram originally sent to Innovative Integration as an example of a basic two pulse experiment that is repeated twice, but with different waveform data in the second run. ABCD represent different waveform data. 15

2.2 A basic nitrogen vacancy example sent to Innovative Integration to show that we will require the DIO timing, as well as delays, to be user defined and accurate. 15

2.3 Basic circuit diagram for a single DIO line on the breakout board. The input originates from the FPGA MDR68 connector, which is terminated by 50Ω at the level converter. An NPN BJT is used to ensure the system can drive a 50Ω load. 19

3.1 Example proof of concept setup of using the X6-1000M to produce phase calibration data. Here the DACs are attached to the mixer's LO and RF ports, and the resulting signal from the IF port is sampled by the ADC. . . 25

3.2 Phase calibration setup in a spectrometer. The directional coupler take a small amount of the post-HPA signal (typically -30 or -40dB coupling) for analysis. This is input to a calibrated IQ mixer for demodulation, extracting the I and Q data separately. The components within the dashed box represent the internals of an IQ mixer. 26

3.3 Example 10GHz ESR spectrometer utilizing the X6-1000M. 27

3.4 Ringdown of the ≈ 8500 Q-factor cavity without any compensation applied. Microwave source was set to 9324.55MHz with an IF of 200MHz. Pulse width was $1\mu s$. 28

3.5 Pulse ringdown after application of a compensation pulse. Microwave source was set to 9324.302MHz with an IF of 200MHz. Initial pulse is $1\mu s$ with the addition of 94ns of inverse phase compensation pulse immediately after. The compensation pulse was set to 97% amplitude to that of the initial pulse. The cursors indicate the approximate start and end of the two pulses. . . . 28

3.6 Vertically zoomed in capture of pulse ringdown with compensation pulse. Here it can be seen that the ringdown is in the noise. 29

3.7 Solid echo pulse sequence. The upper sequence results in an average of zero evolution, while the bottom does not refocus the dipole coupling. Heavily modified version of an image taken from UC Davis ChemWiki [5]. 30

3.8 Oscilloscope trace showing spin echoes generated with identical pulse widths and τ delays. The top trace has the same-phase second pulse, while the bottom includes the $\pi/2$ phase shift. The bottom echo is of larger amplitude as expected. Here, $\tau = 4\mu s$, pulse width is 60ns, and $B_o = 3360G$. 31

3.9 Plot of spin echo data found in table 3.1. Upper trace is the solid echo, lower is the 90-90 echo. 32

3.10 ODMR NV rabi experiment data using the the X6-1000M to gate the RF pulse. 34

3.11 Fast-Fourier Transform of data in figure 3.10. The second peak at ≈14MHz offset indicates that the NV was coupled to a nearby Carbon spin. 35

A.1 Schematic for X6-1000M DIO breakout board made using KiCAD. 43

A.2 PCB layout for X6-1000M DIO breakout board made using KiCAD. This diagram omits the filled copper zones on the front and back for ease of viewing. 44

A.3 Photograph of of revision 3 of the DIO breakout board. The large connector seen at the top is the primary MDR68 connector. The screw-terminal is the power input. All other connectors are SMA. PCB manufacture is APCircuits. Board assembly was performed by myself. 45

B.1 Photograph of the X6-1000M board[16]. The FPGA is located under the heatsink. All the analogue circuitry is located under the shielding on the right. Connectors on the left are used for exposing all the board's communication lines to the enclosure (see figure B.2) 46

B.2 Photograph of the X6-1000M board[16] plugged into the eInstrument[15] enclosure which is used to connect to the board to a PC. Cover is missing to show some of the internals. 47

B.3 Photograph of the X6-1000M board[16] plugged into the eInstrument[15]. Front connectors shown here are the SSMC variety. 47

Chapter 1

Motivation

Microwave (μw) and radio-frequency (RF) control are very important in many areas of spectroscopy. These EM waves are one way to send information into a quantum sample. The more control one has over the precision and accuracy of the outputted waveform, the more complicated the waveform, and thus the algorithm, can be.

The application of Optimal Control Theory (OCT) to a pulse sequence is one such example of a complicated waveform. OCT has improved the robustness of control methods and thus it is favourable to use these techniques. OCT optimized waveforms require fine amplitude and phase control of the μw and RF output in order to accurately produce them. Without precise and flexible modulation schemes one would not be able to successfully output an OCT optimized waveform.

Thankfully, today's modern FPGA based mixed signal development boards provide the specifications required to achieve this. They provide an economic solution to this problem, while also reducing the footprint and offering superior output performance compared to that of traditional test and measurement equipment. Some of these development boards also have additional features such as digital outputs to allow for synchronous operation of peripheral equipment such as switches, lasers, and amplifier gating. This digital control can then result in the elimination of separate dedicated timing equipment, further consolidating an experimental setup.

This thesis proposes and demonstrates the use of an FPGA based development board in order to achieve the high performance as required by OCT. I take magnetic resonance experiments (such as electron spin resonance) as an example to explore this solution.

1.1 Introduction

In pulsed electron spin resonance spectroscopy, the traditional spectrometer design has several limitations when it comes to modern coherent control using OCT; in particular the control of high quality factor resonators.

This chapter will briefly introduce these topics and discuss the high-performance requirements necessary to achieve robust gate performance. It will also discuss general limitations of traditional spectrometer design such as:

- Cost
- Size
- Sample memory
- Acquisition

With these points in mind, we will be able to build a set of requirements for our FPGA-based spectrometer and how they are met in future chapters.

1.2 Optimal Control Theory

In radio-frequency spectroscopy such as nuclear magnetic resonance (NMR) it is favourable to compensate noise processes. This noise compensation can be achieved using adiabatic pulses, composite pulses, and shaped pulses. However, the most general solution will be one that does not necessarily adhere to a simple mathematical function. In fact, the general solution will simply be a list of amplitudes and phases [20].

The means by which Optimal Control Theory (OCT) techniques are applied in order to generate these arbitrary waveforms will not be covered in this thesis. Introductions to this can be found in other sources [3].

Once one has computed the shaped-pulse waveform using OCT, equipment that is physically able to output said waveform is required. This can put fairly strict requirements on the output capabilities of the equipment. This primarily includes the device's amplitude and phase control. The amount of noise compensation that can be achieved through OCT generated waveform is directly tied to one's ability to accurately output the waveform in question.

1.3 High-Q Resonators

One common way of interacting with a quantum system[1] is through a resonance, or tank, circuit. These circuits consist of an inductive and capacitive element placed in parallel with each other. The values of these are chosen such that the system resonates at the desired frequency. The final component of the circuit is a matching capacitor which ensures that the impedance of the entire network is $Z = 50\Omega$. This network can been seen in figure 1.1.

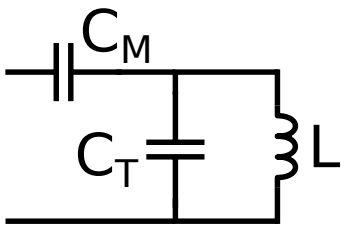

Figure 1.1: Tank circuit block diagram. L and C_T are chosen such that their resonance is that of the desired pulsing frequency. C_M (the matching capacitor) is used to adjust the impedance of the circuit such that $Z_{in} = 50\Omega$.

This circuit converts a pulsed electrical signal into a pulsed magnetic field. This magnetic field is coupled to the quantum system and the desired quantum operation takes place. The magnetic field produced by the quantum system then in turn produces an electric field in the resonance circuit. Typically, transmission and receiving are performed on the same resonance circuit.

Like any resonator, the resonance circuit described above has a quality factor (Q). This is defined as the ratio of stored to dissipated energy in the circuit.

$$Q = \frac{\text{Energy stored}}{\text{Energy dissipated per cycle}} \tag{1.1}$$

One can visualize the Q-factor in this circuit by comparing it to a pendulum. In the pendulum's case, a Q-factor of infinity would correspond to that of an ideal pendulum. Once set in motion, it would never stop. A low Q-factor would be a pendulum that quickly comes to a stop after the initial impulse.

In the case of the above resonance circuit, the Q-factor can be determined by using equation 1.2 along with data from a network analyser.

[1] Popular examples include nuclear magnetic resonance (NMR) and electron spin resonance (ESR).

$$Q = \frac{f_r}{\Delta f}, f_r = \text{resonant frequency}, \Delta f = \text{half-power bandwidth} \tag{1.2}$$

With all factors being equal, the resulting signal strength and signal-to-noise ratio (SNR) scales as $\sqrt{Q}$ [3, 1].

So although a higher Q-factor will result in a higher SNR, it is not without its disadvantages. Once transmission is complete, the same circuit must now act as a receiver. The typical problem with high-Q resonator circuits is that one must wait for this extra energy in the system to dissipate before opening access to the receiver amplifier. This is for two primary reasons. First is that the signal received from the quantum system will be many orders of magnitude smaller than the transmission signals. The second is that the high amplitude of this stored energy can easily overwhelm the receiver pre-amplifier. This can result in the amplifier overloading, forcing one to wait until the amplifier has recovered.

By moving to a higher Q-factor resonator, the ringdown time increases. This can lead to a deadtime greater than that of the characteristic phase coherence time (T_2) of the signal. This in turn lowers the SNR, and with high enough Q-factors, prevents measurement.

One way to deal with these transient signals is to use ringdown suppression pulses at the end of one's pulse sequence. A naive approach to this can be seen in figure 1.2. Here, at the end of the real pulse, a short impulse with the opposite phase is introduced into the system to cancel out the transients. We can once again imagine this like a pendulum. As a pendulum swings back and forth, one could provide a short calibrated impulse against the motion of the pendulum. If the correct amount of power is applied, the pendulum should immediately stop its motion. This is the same case as our resonance circuit. If the correct waveform is applied after our main pulse sequence has completed, it will greatly reduce the ringdown time, and thus reduce the deadtime we must wait before opening the receiver.

In order to implement these ringdown suppression waveforms, one needs equipment that is capable of finely controlling the amplitude and phase of its outputs. More details on these techniques can be found in a variety of sources [3, 2]

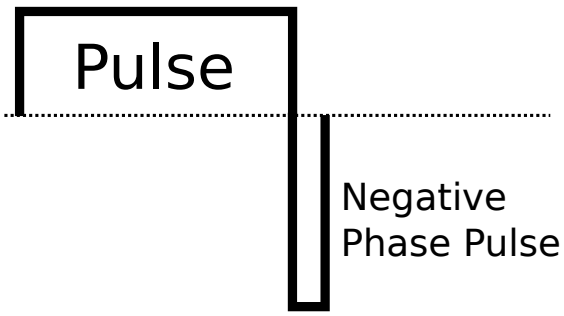

Figure 1.2: Ringdown compensation using a basic negative phase pulse of short duration and high amplitude at the end. An infinite number of solutions exist for the compensation pulse width and amplitude.

1.4 Current Spectrometers

Current typical spectrometer design has several limitations that need to be overcome in order to fully explore the topics in the above sections. First is the issue of on-resonance leakage. A simplified version of the situation is shown in figure 1.3. This consists of a radio frequency (RF) or microwave (μw) source, an arbitrary waveform generator (AWG), and a mixer. The mixer uses the signal from the AWG to amplitude module the high-frequency signal from the RF or μw source.

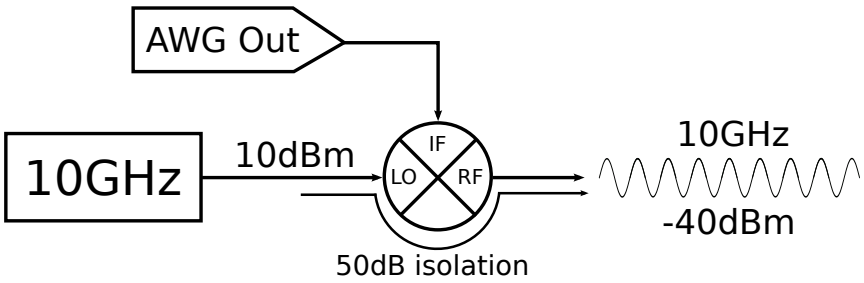

Figure 1.3: Simplified diagram showing LO→RF leakage. Here the LO source is set to 10GHz at 10dBm, which is typical for X-band ESR applications. In this case the mixer has a LO→RF isolation of 50dB, causing a leakage of -40dBm at the RF port.

Due to the analogue bandwidth limitations of typical AWGs used in spectrometers, they are used to generate the envelope of the desired pulse sequence. This envelope is inserted into the "intermediate frequency" (IF) port of the mixer. With the high-frequency source attached to the "local oscillator" (LO) port, the third port (the "radio frequency" (RF) port) produces the multiplicative product of the two inputs. The output of an ideal mixer is that of equation 1.3.

$$RF = LO \cdot IF \tag{1.3}$$
$$= A\sin(\omega_{LO}t) \cdot B\sin(\omega_{IF}t) \tag{1.4}$$
$$= \frac{AB}{2}\left[\cos((\omega_{LO} - \omega_{IF})t) - \cos((\omega_{LO} + \omega_{IF})t)\right] \tag{1.5}$$

In the case where the IF port contains pulse envelope data, we have a situation where the amplitude of the IF port controls how much of the LO port is allowed through to the output RF port.

However, in the real world a mixer does not perform as simple as this. Not only does a mixer produce harmonics, but it also allows a non-zero amount of leakage signal to pass from the LO→RF and IF→RF ports. In this case, the leakage of importance is the LO→RF. Typical isolation values for this range from 35dB to 50dB. Assuming an LO input power of 10dBm this results in a leakage power of -25dBm to -40dBm. Although this seems small, this leakage causes two problems, both of which stem from the fact that the leakage is on resonance with the transitions of the quantum system.

First, this leakage introduces additional errors into any pulse sequence. When it comes to OCT pulses, these errors can prevent the entire pulse sequence from obtaining the desired results. These errors can be accounted for during the generation of the pulses, however this places additional constraints on the algorithm that may be unwanted.

The second problems deals with the receiver amplifier. If our on-resonance leakage is -40dBm, and proceeds through a power amplifier with gain of 40dB, the resulting leakage signal leaving the power amplifier is 0dBm. In a spectrometer, a circulator is placed after the power amplifier. This provides the functionality of directing the high power output to the probe, and the low power response signal from the probe to the receive train. All other directions are isolated. An example of this setup can be seen in figure 1.4. An ideal circulator will have perfect isolation between forbidden signal flow directions. However, in the real world, this isolation value will be finite. A typical circulator might have an isolation value of 20dB. Therefore, with a 0dBm continuous leakage signal entering the

circulator, a -20dBm leakage signal leaves the circulator towards the receive train. Since the signal is on-resonance with the quantum system, it will be of the same frequency of our response signal. This can easily overwhelm the system response and saturate the pre-amplifier. Current techniques to deal with this situation involve switches and blanking the power amplifier. However, both of these require a non-zero amount of time in order to close/turn-off. This lengthens the deadtime before opening the receive switch, reducing the SNR.

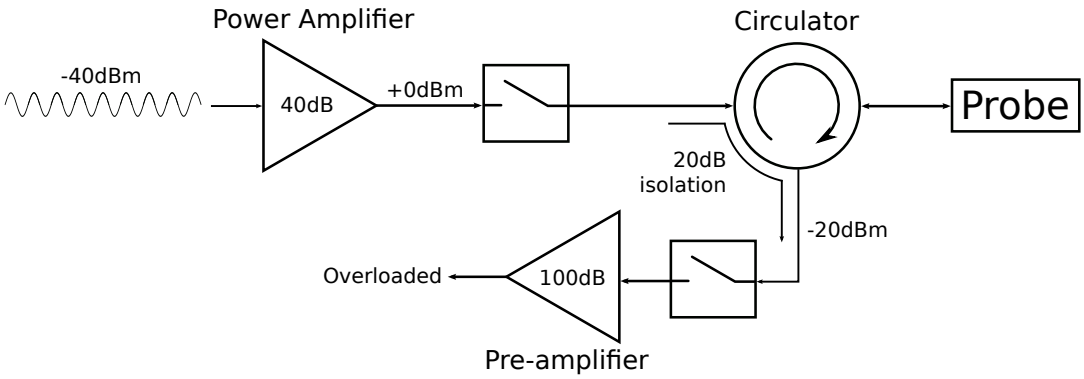

Figure 1.4: Simplified diagram showing circulator leakage. Here the leakage from the mixer (see figure 1.3) is input into the 40dB power amplifier. The circulator allows 20dB isolation leakage through, resulting in a -20dBm signal entering the receive circuitry. This leakage will be consistently present until the switch after the power amplifier is closed.

Other limitations exist within current spectrometers. One of these limitations is the sample memory size of commonly used AWGs. The sample memory is used to store the waveform data. With a typical 16-bit digital-to-analogue converter (DAC), this means that every data point in the waveform uses up 2 bytes of sample memory. Although this may not seem much, it quickly adds up in larger pulse sequences. With an output sample rate of 1GSPs and a waveform lasting 1ms, this would require approximately 1.9MiB[2]. This is a relatively small amount of sample memory and is of no challenge for nearly all AWGs. However, this becomes an issue when the pulse sequence requires long and accurate delays of zero output. This is an issue because the delays must be explicitly stored as an output amplitude of zero. Therefore, if your total pulse sequence, including delays, is one second,

[2]MiB stands for Mebibyte. A Mebi is equal to $2^{20} = 1048576$, which is different from the SI prefix Mega (10^6). This unit was established by the IEC in 2000 for computer science contexts using base 2.

this would use approximately 1.9GiB of sample memory. This is well beyond what most AWGs used in spectrometers are capable of.

Another limitation is the number of outputs. Typical AWGs will only have analogue outputs. A high-performance spectrometer will require digital logic control lines. These lines also need to be synchronous with each other and the main analogue outputs. These lines will control equipment such as lasers and receiver amplifier gating where timing accuracy is critial.

Chapter 2

Solution

In this chapter, the proposed solution to the issues raised in chapter 1 is introduced and discussed.

2.1 Introduction

The chosen solution is to use a Field Programmable Gate Array (FPGA) development board with integrated digital-to-analogue and analogue-to-digital converters (DACs and ADCs), and digital input/output (DIO) lines.

The board selected for this task is the X6-1000M by Innovative Integration[16]. The board features 4 DACs outputs, 2 ADCs inputs, and 32 differential logic DIO pairs. Further information regarding the X6-1000M can be found in section 2.3.

2.1.1 Field Programmable Gate Array

An FPGA is an integrated circuit (IC) whose internal interconnections can be reconfigured as desired after initial manufacturing or "in the field".

The core element of an FPGA is the logic cell. These logic cells typically each consist of a 4-input lookup table (LUT), a full adder, and a D-type flip-flop. The LUT is used to define the custom digital logic required of the cell. They can be configured to perform any digital logic required. By combining together several logic cells, an engineer is able to implement any digital circuit imaginable, all in one IC.

This is an important distinction between FPGAs and micro-controllers/-processors. A micro- is an IC with a set and defined internal interconnection structure. They sequentially execute user provided code and are typically only able to perform one task at a time. Depending on what is running on top of the micro (such as an operating system) there may not be any timing guarantees for the executed code. Contrasting this to FPGAs, one is able to effectively build a digital circuit designed for a specific task within the IC. With this, FPGAs also give the ability to have timing guarantees for various inputs/outputs. You can build a micro-controller within an FPGA, but not the other way around.

These interconnections are not 'programmed' in the traditional sense using a language such as C. Instead, a Hardware Description Language (HDL) is used. The two most popular HDLs are VHSIC Hardware Description Language (VHDL, where VHSIC in turn stands for Very-High-Speed Integrated Circuits) and Verilog. HDLs are very different to that of software programming languages (C, Python, Perl, etc) as they are used to describe the behaviour of the desired digital circuit.

FPGAs are commonly used in applications requiring strict timing, high performance per watt computations, and/or massively parallel algorithms. Some examples include code cracking [23, 4], bitcoin mining [30], and development of high-speed communication buses.

These properties of an FPGA make it perfect for use in a spectrometer system. More information about FPGAs in general can be found at a variety of sources [27, 17].

2.2 Why Now?

Now is the perfect time to pursue a project of this calibre. Modern FPGA, DAC, ADC, and personal computers (PCs) have progressed to the point where we are able to adequately address the issues raised in chapter 1.

Even just a few years ago one would not have been able to economically assemble the required components to achieve the desired performance. Sample memory of digital storage oscilloscopes were in the low MiB range, DACs were far too slow to directly synthesize any high frequency waveforms, and even PC specifications were not sufficient in the early 2000s to cost-effectively run the pulse finding algorithms associated with the subjects discussed in chapter 1.

In addition, we now have the computational power to run the algorithms associated with OCT and high-Q resonators. This makes it the perfect time to re-evaluate how current spectrometers are designed, and improve on them so that we may continue to explore these developing topics.

2.3 Innovative Integration X6-1000M

Instead of designing an FPGA board from scratch, a third-party solution was chosen that met our requirements. After searching through various manufacturers, the X6-1000M by Innovative Integration[16] was selected. This board features a Xilinx Virtex-6 FPGA[31], 2 Texas Instruments DAC5682Z DACs[12], 2 Texas Instruments ADS5400 ADCs[11], a PCI-Express v2 x8 high-speed interface, and 4GiB of on-board waveform memory.

Each of the two DACs contain two outputs. With one output enabled, the DAC is capable of speeds up to 1GSPs (giga-samples per second). With both enabled, the output speed drops to 500MSPs. Therefore, the board has either 2 outputs at 1GSPs each, or 4 outputs running at 500MSPs each. This is entirely user configurable at runtime and does not require re-synthesis of the FPGA configuration.

The ADCs each run at 1GSPs maximum and sample directly into the on board memory. By default under the stock FPGA configuration they can only be turned on after the DACs have completed outputting their waveforms. This is a limitation of the stock firmware as it is desirable to be able to start the acquisition at any point in time.

The X6-1000M exposes 32 low voltage differential signalling (LVDS) digital input/output (DIO) pairs. By default, these DIO lines are not synchronous with the DACs/ADCs. This is an additional limitation of the stock firmware.

FPGA	Xilinx Virtex-6
Speed Grade	-1
Size	31M gate equivalent
Configuration	JTAG, FLASH
Interface	PCI-Express v2 8x
DRAM Size	4GiB (4x1GiB banks), 1GiB per hardware DAC/ADC
DRAM Rate	Up to 5.2GiB/s sustained transfer rate per bank
DRAM Clock	333MHz

Table 2.1: X6-1000M Specifications[16]

2.4 Stock FPGA Configuration

The X6-1000M comes with a stock configuration that is used to demonstrate the basic functionality of the various components. The board also ships with several C++ example applications to interact with the stock configuration.

Inputs	2
Range	1Vpp
Coupling	DC
Impedance	50Ω
Device	Texas Instruments ADS5400
ADC Resolution	12-bit
ADC Sample Rate	1MHz to 1GHz
Bandwidth	25GHz

Table 2.2: X6-1000M Analogue Input Specifications[16]

Outputs	4 channels at half rate, 2 at full
Range	1Vpp
Coupling	DC
Impedance	50Ω
Device	2X Texas Instruments DAC5682Z
DAC Resolution	16-bit
DAC Sample Rate	1MSPs to 1GSPs

Table 2.3: X6-1000M Analogue Output Specifications[16]

The stock configuration allows for the following:

- Outputting standard waveforms (sine, square, triangle, etc) generated either by the C++ program and loaded onto the board, or directly generated by the FPGA
- Acquisition of waveforms through the ADCs
- Asynchronous digital input/output line control
- Calibration settings
- Internal or external trigger settings
- Internal or external clock source settings
- Clock PLL value (up to 1GHz)

However, with these capabilities comes with a long list of limitations. These shortcomings prevent the X6-1000M stock configuration from being usable in a spectrometer.

- DACs may only be enabled once per trigger, and they all must be enabled together at the same time. This means that any delays in the waveform must have zeros explicitly stored. As mentioned in chapter 1, this is a disadvantage of conventional AWGs and is something that needs to be avoided.
- ADCs are enabled after DACs have completed, with no guarantee as to the exact clock that they do turn on.
- Waveforms can only be played out once per trigger. In order to re-run, the waveform must be re-uploaded. This is not acceptable for any experiment that requires many runs for statistical analysis (example, nitrogen vacancy systems[1]).
- Asynchronous DIO line control. With no guarantees as to when, or for how long, any DIO line will be enabled, this makes them useless for nearly all applications.

In addition to the above, the included C++ control programs are rather limited in usefulness. They are pretty much just limited to doing basic testing with the hardware and do not lend themselves to inclusion in any sort of experiment automation solution.

These limitations make the X6-1000M stock configuration useless in a spectrometer setting. In fact, since you are unable to interact with the board from a separate programming environment, it would actually be a worse option than the standard AWG. Thankfully, since FPGAs can be freely re-configured, a custom configuration file and supporting PC software could be made.

2.5 Custom FPGA Configuration

With the capabilities and disadvantages of the stock configuration known (see section 2.4), I met with as many members of our research group as possible to build a list of desired functionality. The main requirements that everyone would need include:

- Ability to enable/disable hardware elements (DACs, ADCs, DIOs) as desired. This will allow us to turn off the DACs during times of zero output while still keeping track of elapsed time, preventing the unnecessary storage of zeros in the waveform memory.

[1]Nitrogen Vacancy (NV) systems can require over 50 000 runs for a single data point.

- All hardware elements should be synchronous with each other in order to ensure that experiments are consistently repeatable and outputs are accurately generated. The accuracy of this should be directly proportional to that of the board's clock source, while the resolution should be on the order of nanoseconds.

- The option to repeat a waveform an arbitrary number of times without having to reload it from the computer. If a specific experiment calls for many repetitions of the same parameters, it is desirable to avoid having to continuously reload the same data onto the board. Data transfer and board initialization takes a finite amount of time, and when an experiment calls for 50 000 repetitions, this delay can prevent the experiment from completing in a reasonable amount of time.

Innovative Integration provides the required framework in order to synthesize your own FPGA configuration using either Xilinx ISE or Mathworks Simulink. However, FPGA synthesis is a non-trivial task, especially at high clock-rates. As mentioned in section 2.1.1 an FPGA configuration routes the signal flow inside the IC. At higher clock frequencies, the designer has to take into account many effects, such as cross-talk, and ensuring that grouped signals arrive at the same time. Writing HDL at this speed requires years of experience. This is something that our group just does not have, and thus the decision was made to utilize the experience at Innovative Integration in order to build our custom configuration.

Here, I managed the communications between the II engineering team and our group members. This involved managing the expectations of the group members, while also trying to communicate to the II engineers what exactly we are looking for. Some examples of diagrams that were made to help convey our requirements can be found in figures 2.1 and 2.2.

Since an FPGA is a digital system, bit-width for various parameters needed to be agreed upon. It is impossible to make the system infinitely long, so a balance had to be reached between what the group wanted from the system, and what the engineers thought was reasonably possible. Some of these parameters include maximum hardware delay, number of sequence repetitions, and pulse width. A summary of every parameter, its bit-width, and corresponding maximum time in seconds, can be found in table 2.4. All other relevant parameters for the hardware can be found in table 2.5

Along with the custom configuration, various other pieces of software were commissioned from II. This software included the following:

- C++ example software

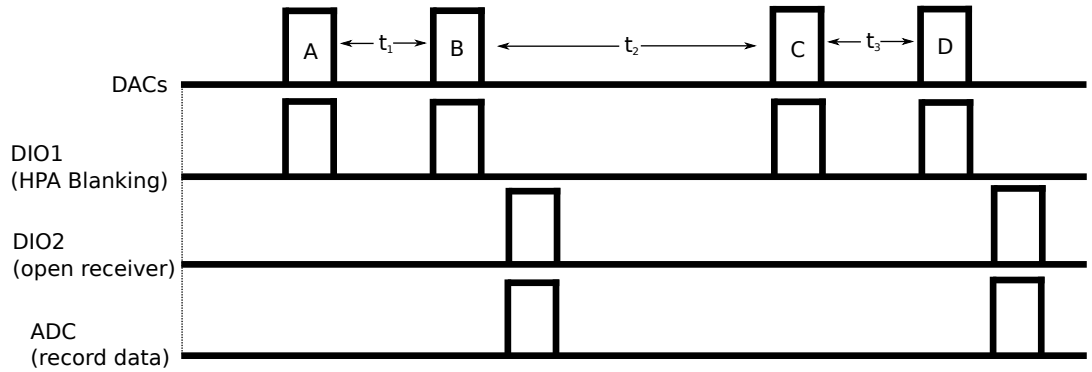

Figure 2.1: Diagram originally sent to Innovative Integration as an example of a basic two pulse experiment that is repeated twice, but with different waveform data in the second run. ABCD represent different waveform data.

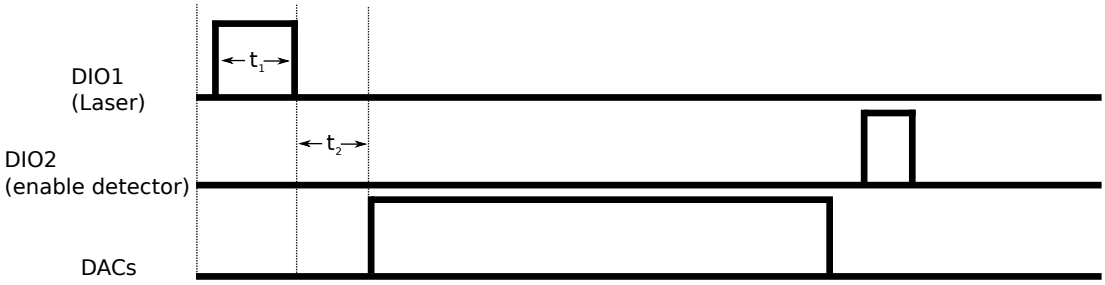

Figure 2.2: A basic nitrogen vacancy example sent to Innovative Integration to show that we will require the DIO timing, as well as delays, to be user defined and accurate.

- Windows 64-bit DLL file
- C++ example using the DLL

The C++ example software serves as a way to check the functionality of all the new features. However, similar to the stock PC software and stock configuration, it is not part of the long term plan for the FPGA as it cannot be integrated into some sort of larger software solution for an experiment.

Parameter	Bit-width	Real world numbers
Delay	40-bith	˜1.2h
Width	40-bit	˜1.2h
Frame length	44-bit	˜19.5h
Frame count	32-bit	˜4.3 billion

Table 2.4: List of bit-widths for various parameters in the custom FPGA configuration by Innovative Integration. Delay is the time before pulse is enabled, width is how long it stays on for (see t_1 in figure 2.2), and a frame is equal to a signal "experiment". Therefore, frame length is the maximum time for one experiment to run, while frame count is the number of times a frame can be repeated.

Parameter	Value
FPGA core clock	250MHz
Resolution	1 clock (4ns)
DAC & ADC rate	1GSPs
Pulses per hardware line per frame	32

Table 2.5: List of miscellaneous figures related to the custom FPGA configuration by Innovative Integration. Resolution is key as it defines the minimum step one can adjust the delay and width parameters from table 2.4.

This is where the Windows 64-bit DLL (dynamically loaded library) comes in. The role of this piece of software is to provide a bridge between a separate programming language and the FPGA communication libraries (which are written in C++). This DLL allows languages/environments such as Python, MATLAB, and LabVIEW to directly interact with the X6-1000M custom configuration.

2.5.1 Limitations

This custom configuration by II is not without its limitations. It can be very difficult to cover every base and get everything perfect on the first iteration of a piece of software. Also, as the board gets deployed throughout various labs within our group, and thus exposing it to a larger variety of environments and users. As more people use it, more errors and concerns become visible, which will eventually be used as feedback for a second version of the FPGA configuration and PC side software.

Some limitations that have been discovered include:

- Lack of error propagation from the DLL. This means that if something goes wrong (for example, invalid input parameters into the DLL methods, communication issues, etc) the user is unable to figure out what happened. This makes debugging and initial development difficult and frustrating as errors must be discovered and debugged through trial-and-error.
- There currently exists an error that causes all of Windows to freeze and requires a hard-reset in order to solve.
- The limitation of 32 pulses per hardware device per frame was originally agreed upon during our conversations, but has since been found to be insufficient.
- Lack of "operation complete" notification from the FPGA and DLL. This is of particular importance when loading the waveform into the on-board memory. It is very useful on the PC side to know when an operation as critical as this has completed.
- Poor software trigger behaviour. Currently, if set to software trigger, the trigger event occurs at the exact time as defined by the "software trigger delay" parameter. The counter for this starts when waveform transfer also begins. This means that if the waveform takes longer to load than the trigger delay is set for, the output will still begin and cause a large number of output errors. Of course, if a delay larger than required is used then time is just being potentially wasted.
- External trigger phase errors. It has been observed that when using the external trigger, the occasional run has a 1ns or 8ns delay introduced to any of the hardware channels. However, we have also noticed recently that this also occurs with the default FPGA configuration. The firmware team at II are currently investigating this issue.

Going into the future, these are all items that will need to be addressed in order to produce the working solution. However, none of the above prevent the X6-1000M from being a superior solution to that of current setups.

2.6 DIO Breakout Board

The X6-1000M exposes 32 differential pair digital input/output lines via the rear "mini d-subminiature ribbon 68 pin" (MDR68) connector. The custom FPGA configuration overrides the default signalling behaviour of these DIO lines, and instead exposes 16 single

ended DIO lines. Those 16 independent lines signal using 2.5V high and 0V low voltage levels. They are directly connected to the FPGA IC, and are capable of driving a 50Ω load.

This situation presents two problems. The first is the 2.5V signaling level. Nearly all laboratory instruments will call for a 5V high trigger signal. Equipment using a BJT input (5V-TTL) might be able to trigger off the 2.5V signal, but a CMOS input (which requires a $>$ 3.5V input) will not be able to. Because of this, the 2.5V FPGA DIO signal must be converted to 5V, and still be capable of driving a 50Ω load. The second problem is the connector. An MDR68 connector is very dense and needs to be broken out into more convenient connectors. Since most equipment has either BNC or SMA connectors for their trigger ports, SMA connectors are the optimal choice for a breakout board.

The first section of the breakout board involves level translation. This is done using two TI SN74LVC8T245 8-bit dual supply translating transceiver[14]. In addition, each DIO line is terminated by a 50Ω load as required to minimize reflections.

These level converters are not capable of directly driving a 50Ω load on their own. The recommended way, as per *The Art of Electronics* by Horowitz and Hill, is to use an NPN bipolar-junction transistor (BJT)[22]. By attaching the output of the level converter to the base of the transistor, a 5V power supply to the BJT collector, and the SMA connector to the emitter, the system is capable of driving a 50Ω load at 5V. As an extra precaution, a 10Ω resistor is placed in series with the power supply and the BJT collector. This is to protect the transistor and power supply against short-circuits occurring after the SMA connector. This reduces the voltage at the load to $\approx 4V$, so the power supply is increased to 6V, leaving 5V at the load. The transistor used is an NXP PBSS2515VS[24] 15V low $V_{CE(sat)}$ NPN double transistor. An example of this can be seen in figure 2.3.

Although many pieces of equipment in a laboratory use TTL, not all do. One logic standard that was once very popular but more recently fallen out of use is emitter-coupled logic (ECL). ECL is a high-speed BJT logic family which avoids the saturated (fully on) region of transistor operation to avoid its associated slow turn-off behaviour[26]. By using small voltage swings (0.8V), the transistors are able to change state quickly and operate at a high-speed.

Every ECL gate continuously draws large amounts of current, and therefore dissipates a lot of heat. This means that ECL gates can only be so closely packed, require huge amounts of power, and plenty of heat dissipation. In addition, as BJT technology has improved, lower voltage versions of TTL have been developed such as LVTTL which uses 3.3V power supplies instead of the standard 5V for TTL. All the improvements to the TTL family have pushed ECL away from every day use. However, improvements continue to

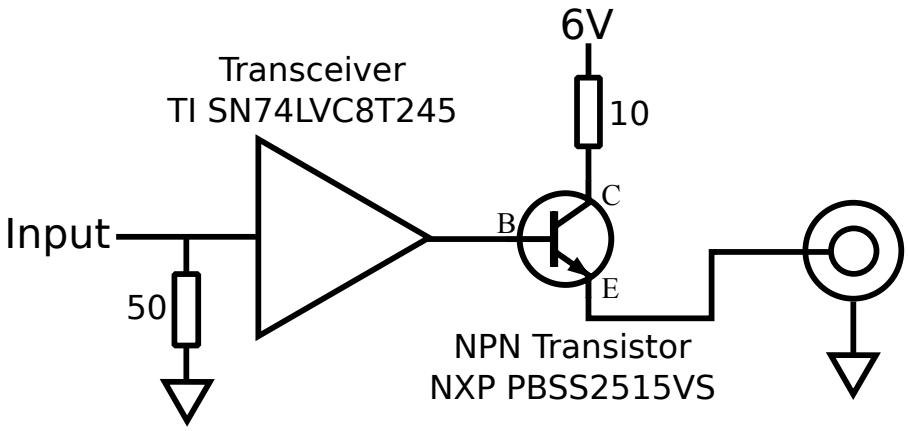

Figure 2.3: Basic circuit diagram for a single DIO line on the breakout board. The input originates from the FPGA MDR68 connector, which is terminated by 50Ω at the level converter. An NPN BJT is used to ensure the system can drive a 50Ω load.

be developed for ECL. Modern ECL gates are designed with sub-micron transistors with low parasitics[25]. This reduces the power consumption and allows for faster operation. Today's ECL gates can be toggled at greater than 3GHz.

Standard ECL requires the use of a -5.2V power supply, with logic levels of -1.75V for low, and -0.9V for high[13]. This means that the breakout board as currently designed is unable to interface with equipment using ECL trigger inputs. Additional TTL-to-ECL adapters would be required to interface the two.

Other variations of ECL exists, including positive ECL (PECL) where the power supply is 5V, where high is 4.2V and low is 3.4V. If the equipment uses PECL, the breakout board should be able to interface with it, depending on the system behaviour when the input is 0V.

Another situation is where the trigger input jack of the instrument is floating instead of referenced to earth's ground. If this is a necessary condition for the equipment, an isolation buffer will be required. The breakout board's ground is directly connected to that of the X6-1000M, which is in turn connected to the PC's earth referenced ground. Equipment that has a metal chassis and takes mains power is going to have the chassis tied to earth's ground for safety reasons. This means any input/output connectors that are mounted to the chassis will also be tied to earth's ground. A common piece of equipment where this is not the case is frequency generators. If electrical isolation is important to the application,

care must be used when connecting the breakout board.

The entire board design was done using the open source electronic design automation (EDA) software package KiCAD[19, 29]. The software is available for Windows, Linux, and OSX, and is free as in gratis and libre[2]. The schematic and board layout can be found in figure A.1 and A.2, respectively.

2.7 Python Communication Package

As mentioned in section 2.5, the custom FPGA configuration from Innovative Integration came with several pieces of PC side software. One of those is a Windows 64-bit DLL, which is used to bridge the gap between one's programming language of choice and the II X6-1000M communication libraries.

Through the work of two group members, Chris Granade and Ian Hincks, an entire Python[6] module was created to ease user interaction with the DLL. This abstracts the user away from having to deal with Windows DLL specific issues and just focus on interacting with the X6-1000M. In addition, this Python module has allowed us to implement our own error handling for issues that we have come across and debugged.

Python was chosen over other options (for example, MATLAB) for several reasons. It is a very simple language (on the surface) which helps keep the barrier of entry as low as possible for non-programmers to start using the system. Python is also open source, which means (unlike MATLAB) it has no licensing requirements.

Packages such as NumPy[21], SciPy[18], and Matplotlib[9] provide similar functionality to MATLAB. These packages are programmed at a low level, giving Python near-C speeds for array and matrix computation.

Here is an example using the package to load configurations and waveform data onto the board:

```
>>> import x6
>>> x = x6.X6()
>>> x.open()
>>> x.load_configuration('file.pulse','Compiled_Pulse')
>>> x.preconfigure()
>>> x.start_streaming()
```

[2]In English there is two meanings of the word "free". KiCAD is both "for zero price" (gratis) and "with little or no restriction" (libre) [28].

Therefore, we have 6 lines to go from a new Python kernel to waiting for our trigger event. By breaking this down, we can see what happens at each step:

```
>>> import x6
>>> x = x6.X6()
```

These first two lines import the package and create our object representing the X6-1000M.

```
>>> x.open()
```

The third line opens the connection to the board. This usually takes a second or two. A good sanity check to run after opening connection is x.logic_version() which returns basic version information for the board and its configuration. Reading the version information is in no ways required for operation.

```
>>> x.load_configuration('file.pulse','Compiled Pulse')
```

This line loads the file containing all of the desired settings for the board. This includes which hardware elements are enabled, trigger settings, repetition behaviour, and file locations to the waveform data. A single ".pulse" file has support for multiple groups of settings, where the desired one is selected by the second parameter (this example is "Compiled Pulse"). This helps keep the number of files in the project directory to a minimum.

```
>>> x.preconfigure()
```

Preconfiguring is the process during which the various settings of the experiment are uploaded onto the board. The important thing to note about this step is that the DACs will produce bursts of high-frequency output, which can damage sensitive equipment. For example, if the FPGA is attached to an amplifier, it could damage downstream equipment. This is not a limitation of the board as the user should be taking the steps necessary to protect their equipment. Proper use of limiters and amplifier gating is always recommended.

```
>>> x.start_streaming()
```

The last line starts waveform transfer as well as putting the FPGA into a wait-for-trigger state. If using software trigger, it will occur after the delay as specified in the configuration file ('file.pulse').

The Python package also contains other useful tools, such as the ability to convert NumPy arrays into the required file format. These are very straight forward to use, and

they make the process of generating the waveform data to preparing it for the board all self-contained within the Python environment. An example generating the files for a 200MHz pulse, containing both I and Q waveforms can be seen below:

```
>>> import numpy as np
>>> import x6.process_waveform as pw

>>> f = 200*10**6 # 200MHz
>>> w = 2*np.pi*f
>>> t_step = 10**-9 # 1ns DAC resolution
>>> length = 1000 # 1000 samples at 1ns resolution = 1us
>>> t = np.linspace(0, length-1, length)
>>> wf1 = np.sin(w*t*t_step)
>>> wf2 = np.sin(w*t*t_step + np.pi/2)

>>> wf1 = wf1 * (2**15 - 1) # Adjust to full 16-bits
>>> wf2 = wf2 * (2**15 - 1)

>>> pw.waveform_to_velo(
        active_channels = [True,False,True,False],
        output_filename = 'waveform.velo',
        waveform0 = pw.Waveform(wf1),
        waveform2 = pw.Waveform(wf2)
        )
```

The key is the last line of Python, where the desired DACs to be enabled are specified, and the NumPy arrays containing the waveforms are loaded. The waveform_to_velo() method takes these inputs, along with a desired output filename, and generates the correct "velo" file as required by the X6-1000M DLL.

In addition, Granade and Hincks made an extra effort for those that want to quickly convert current PulProg based experiments over. They implemented a basic graphical interface and a PulProg interpreter to give the user the option of completely avoiding Python. PulProg, or Pulse Program, is a basic language used in the Bruker Topspin XWin environment on NMR consoles.

PulProg is used to do things like define frequencies, pulse widths, delays, and other variables. The x6 Python package includes tools to read these files and perform the same actions. Not every feature of the language is supported, but it allows someone who is more comfortable with PulProg to make basic pulse programs.

Since the majority of users who wish to use this PulProg interpreter will be doing so because they are not comfortable interacting directly with Python, there is also a graphical user interface where the user can load their PulProg code and view a visual rendering of their pulse sequence. From here the program can be loaded onto the FPGA and executed.

Naturally, this option is more limiting compared to just using Python directly as it prevents you from using any of the tools mentioned above (NumPy, etc).

Chapter 3

Implementations

3.1 Phase Calibration

As mentioned in chapter 1, many of the potential applications of this FPGA system involve fine phase control. However, most equipment used in a spectrometer do not have a flat phase response. This means that the errors introduced by the transfer function of the equipment after the DACs can prevent the desired waveform from reaching the sample. A distorted waveform will not perform the desired action on the quantum system.

The biggest offenders of this is the power amplifier, which is primarily non-linear in amplitude, and the resonator, which is non-linear in phase as a function of frequency. To compensate for these errors, a look-up matrix of calibration data can be constructed by sampling the signal just before the probe. Using the X6-1000M, a test-signal is generated of a sweep in phase and amplitude, and a directional coupler is used to extract a small amount of the signal. This is read by the ADCs.

As a small proof of concept, the following experiment was run. Using NumPy, two identical sine waves are generated with a variable phase between them. These signals are input into the LO and RF ports of a mixer and beat against each other (figure 3.1). This produces a DC and $2f_o$ signal coming out of the IF port. By sampling this signal and taking the average, the DC component can be found.

A perfect phase response will produce a linear graph with an x-intercept at $\frac{\pi}{2}$ phase difference. Deviations from this ideal x-intercept correspond to a constant phase phase delay (cable lengths, mixer phase imbalances) for one of the channels.

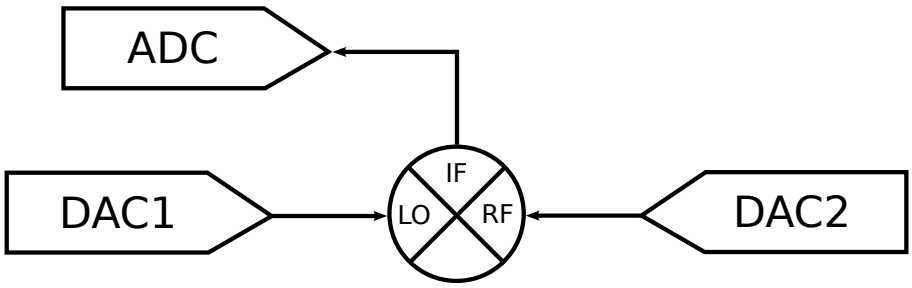

Figure 3.1: Example proof of concept setup of using the X6-1000M to produce phase calibration data. Here the DACs are attached to the mixer's LO and RF ports, and the resulting signal from the IF port is sampled by the ADC.

Figure 3.2 shows the main components associated with this phase calibration for an actual spectrometer. A sample signal is coupled out after the high-power amplifier (HPA) and is fed into a small network of known calibration. The attached PC can then do analysis on the sampled data, generating a calibration matrix for the entire spectrometer.

3.2 ESR Spectrometer

One application for the X6-1000M is within a pulsed electron spin resonance (ESR) spectrometer. The board, along with its custom configuration and PC side software, are the focal point of a new home-built ESR spectrometer currently being constructed. The general setup of this spectrometer, along with notes to the specific components selected can be found in figure 3.3.

This design has the board generating both the I and Q waveforms for the IQ mixer directly with a carrier frequency of 200-500MHz. This leaves the leakage off resonance. The other important aspect is the amount of receiver gain. The gain is spread over two frequencies in order to prevent amplifier oscillation. First, a 10GHz pre-amplifier is used, then the signal is mixed down to the same center frequency as originally directly generated by the X6-1000M (the intermediate frequency), and finally another amplifier stage is applied prior to acquisition. If 100dB+ of amplification were to be applied all at the same frequency, the amplifiers would oscillate. This is due to a small amount of leakage from the output that gets coupled back into the input. If the amplification exceeds the feedback isolation, the amplifiers saturate.

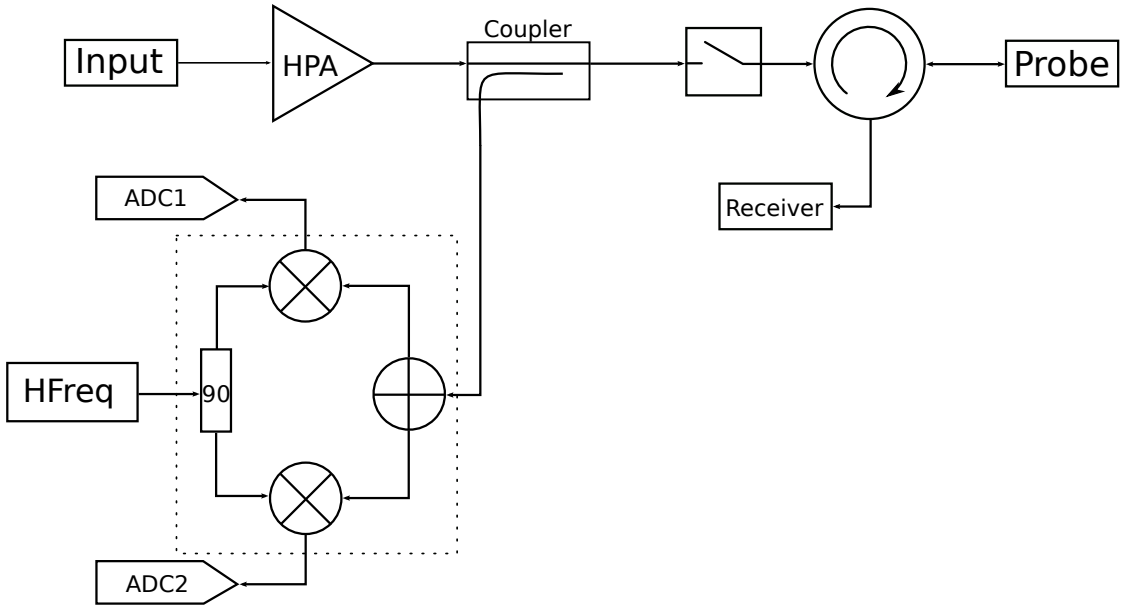

Figure 3.2: Phase calibration setup in a spectrometer. The directional coupler take a small amount of the post-HPA signal (typically -30 or -40dB coupling) for analysis. This is input to a calibrated IQ mixer for demodulation, extracting the I and Q data separately. The components within the dashed box represent the internals of an IQ mixer.

3.2.1 Ringdown Compensation

As discussed in section 1.3, one of the potential applications of the X6-1000M is for use with high-Q resonators. To demonstrate this, the spectrometer was used in conjunction with a Varian E-231 rectangular cavity (Q≈ 8500). This cavity has been previously found to require a deadtime of $1.2\mu s$ to allow the pulse ringdown to decay below the noise floor[2]. The inhomogeneous phase coherence relaxation time of the sample has also previously been found to be $T_2^* \approx 250ns$[2]. This means that ringdown compensation is required in order to directly observe the free-induction decay of this sample in our high-Q cavity.

In order to cancel the ringdown of the cavity, the simple approach of using a relatively strong, negative phase pulse was used. The pulse length and amplitude, as well as the microwave source frequency, were adjusted in order to minimize the ringdown. This resulted in superior ringdown compensation to that which was achieved prior to the addition

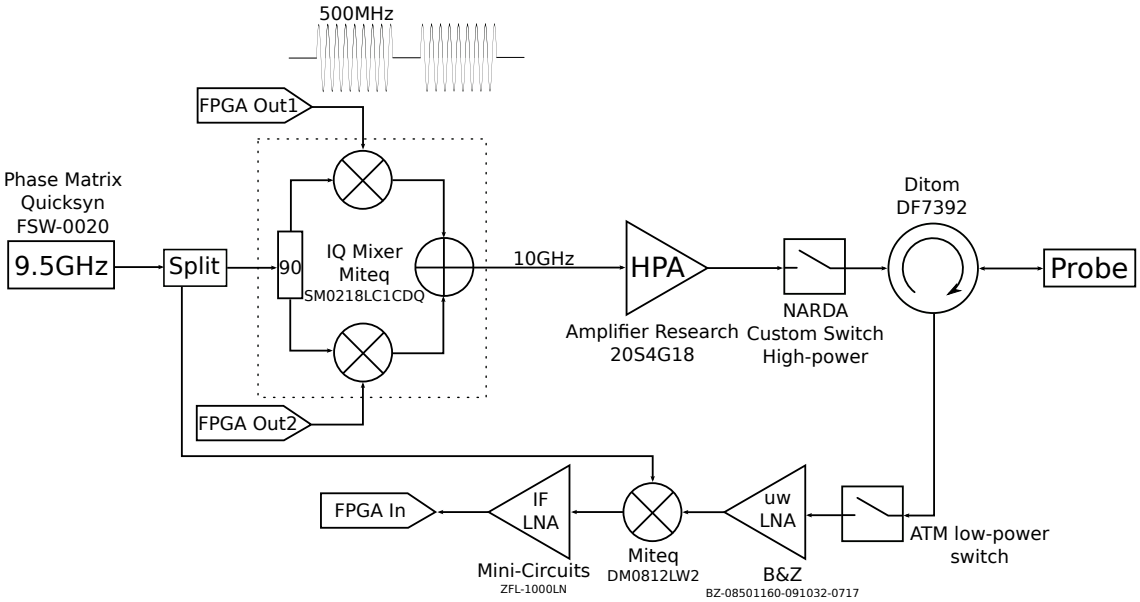

Figure 3.3: Example 10GHz ESR spectrometer utilizing the X6-1000M.

of the FPGA board[2]. Oscilloscope traces showing the effectiveness of the compensation pulse can be seen in figures 3.4, 3.5, and 3.6. These figures show far better ringdown compensation than previously achieved, as well as the elimination of the on-resonance leakage allowing for immediate measurements in the noise floor.

However, a free-induction decay (FID) signal was unable to be obtained due to the resonance frequency drift of the cavity. Over the course of just a few hours, the resonance had drifted by over 2MHz, rendering the ringdown compensation parameters used no longer valid.

Future work for this involves obtaining an FID. To do this, either a newer and more stable cavity is required, or a software feedback loop is required to continuously adjust for the drifting resonance frequency. This control loop could be completely automated into some sort of calibration sequence. This could also include the phase calibration as discussed in section 3.1.

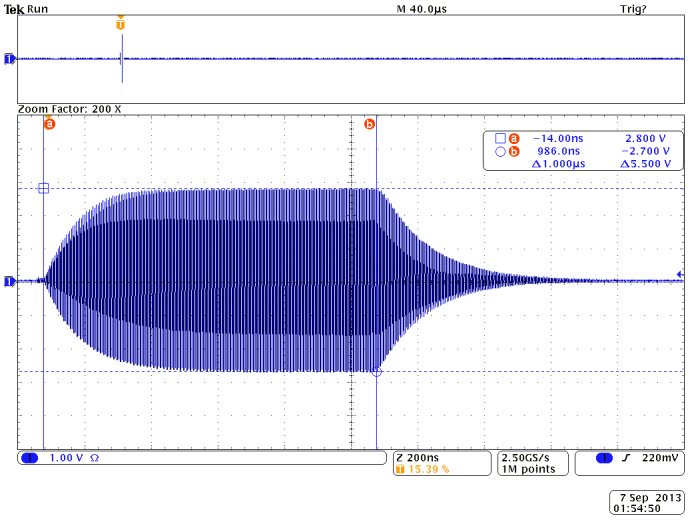

Figure 3.4: Ringdown of the $\approx$ 8500 Q-factor cavity without any compensation applied. Microwave source was set to 9324.55MHz with an IF of 200MHz. Pulse width was $1\mu s$.

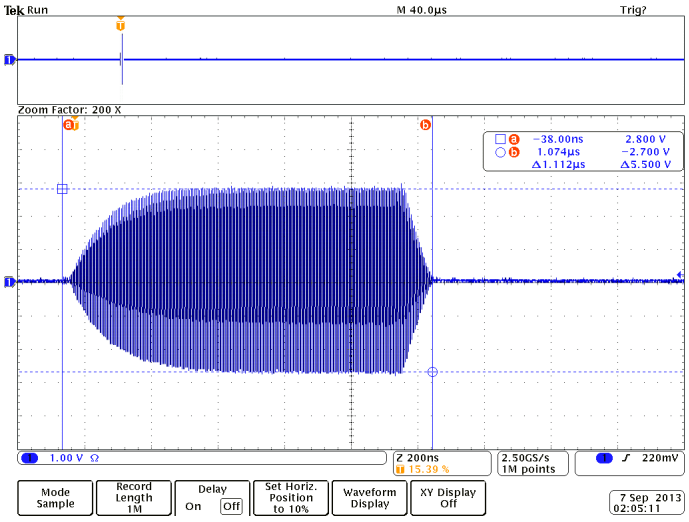

Figure 3.5: Pulse ringdown after application of a compensation pulse. Microwave source was set to 9324.302MHz with an IF of 200MHz. Initial pulse is $1\mu s$ with the addition of 94ns of inverse phase compensation pulse immediately after. The compensation pulse was set to 97% amplitude to that of the initial pulse. The cursors indicate the approximate start and end of the two pulses.

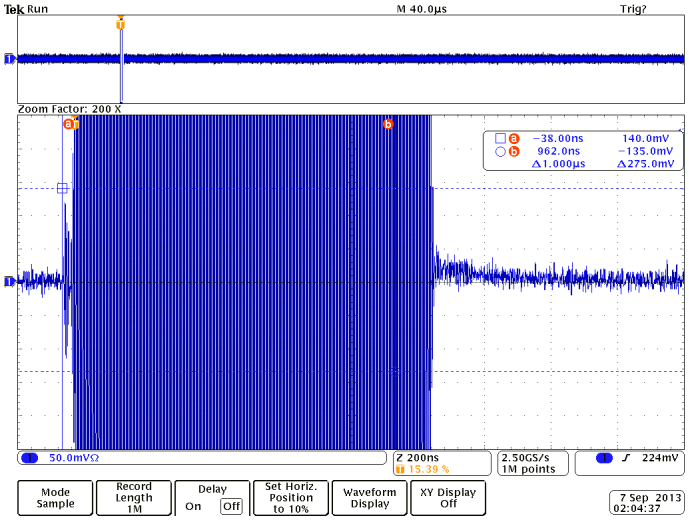

Figure 3.6: Vertically zoomed in capture of pulse ringdown with compensation pulse. Here it can be seen that the ringdown is in the noise.

3.2.2 Solid Echo

An alternate detection scheme to directly observing the FID is to observe a "spin echo" [8]. In this experiment, two pulses of nutation angle α and β with phases ϕ_1 and ϕ_2, respectively, are applied to the quantum system, separated by a variable delay τ.

An intuitive explanation for the formulation of a spin echo may be obtained by considering the case where $\alpha = 90^o$ and $\beta = 180^o$. The first pulse tips a portion of the equilibrium spin magnetisation into the XY plane of the Bloch sphere. The spins then begin to precess under the force applied by a large static magnetic field. These spins precess according to their local field, which varies over the ensemble due to spatial inhomogeneities of the applied field. This leads to a distribution of Larmor precession frequencies that de-phases the net sample magnetization with a time-constant T_2^*. The second pulse reflects the spin state about an axis in the XY plane such that further precession reverses the phase accumulation of the first delay period. After a second delay of τ, the spins re-phase, leading to an echo.

Mathematically, a spin echo may be understood through the use of Average Hamiltonian Theory (AHT)[7], which gives the effective Hamiltonian dynamics to varying order in the toggling frame (ie time-dependent interaction frame, coined toggling frame by Waugh [7]) of a series of pulses. The average Hamiltonian is normally given over a cycle, such that the overall toggling frame operation is the identity. For a spin echo, a cycle consists of 3 delays

and 2-π pulses. The interaction frame Hamiltonian during the first and final delays is given by $\frac{1}{2}\Delta\omega\tau\sigma_z$. The effect of the π pulses is to invert the effective Hamiltonian during the middle delay period of length 2τ. To zeroth order over the cycle, the average Hamiltonian is then zero and the coherent evolution of the spins is given by the identity operation. The spin state at the time of the echo peak will be proportional to the initial magnetization immediately after the 90 degree pulse, with signal loss due to relaxation processes.

A limitation of this spin echo sequence is that it does not refocus de-phasing due to the dipolar coupling strengths between members of the ensemble. The dipole coupling Hamiltonian is $H_D = w_D(\sigma \cdot \sigma - 3\sigma_Z\sigma_Z)$. In a solid sample, these dipolar couplings lead to a shortening of the apparent T_2 of the spin sample, often preventing acquisition of the spin echo. In this situation, from the standpoint of AHT, the zeroth order average Hamiltonian is not identity except when certain conditions are met for α, β, ϕ_1, and ϕ_2. When $\alpha = \beta$ and $\phi_1 = 0^o$, $\phi_2 = 90^o$, the zeroth order average Hamiltonian of the dipolar interaction becomes identity and an echo is observed. This can be seen in the upper diagram of figure 3.7 where the evolutions average out to zero. This echo is known as a 'solid echo'. The solid echo refocuses both static field inhomogeneities and eliminates de-phasing due to dipolar couplings, leading to an echo that is equal or larger than that of a spin echo.

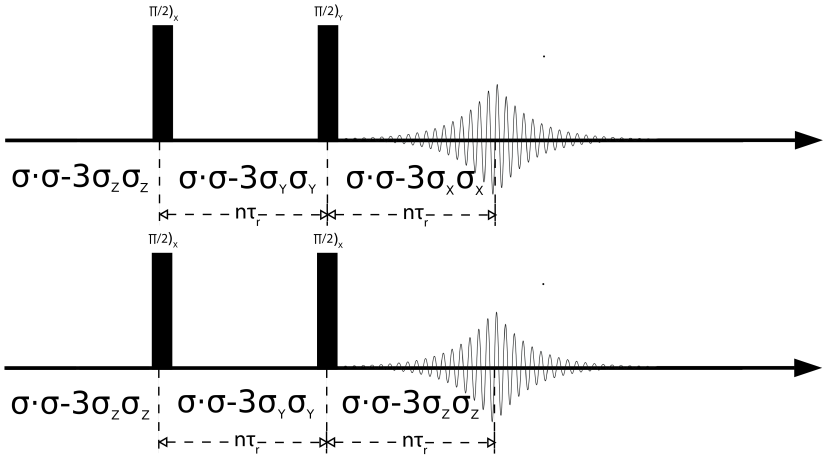

Figure 3.7: Solid echo pulse sequence. The upper sequence results in an average of zero evolution, while the bottom does not refocus the dipole coupling. Heavily modified version of an image taken from UC Davis ChemWiki [5].

Use of the X6-1000M allows us to conveniently implement phase modulation of our pulses, permitting an investigation of the difference between a spin echo and solid echo in an irradiated quartz sample inserted in the high-Q cavity.

Two approaches to the solid echo was performed using the FPGA. The first involved setting both pulses to have the same phase, while the second had the second pulse include a 90^o phase shift. Both of these required accurate phase tracking of the system during the delay period τ.

The experiment involved using pulses 60ns long ($\alpha = \beta \approx 90^o$), while varying τ from $2\mu s$ to $20\mu s$. This was done for both phase variations of the second pulse. It is expected that the sequence with the phase shift will produce an echo signal of stronger intensity and slower decay.

An example oscilloscope trace of this behaviour can be found in figure 3.8. The difference in echo amplitudes is only a small amount due to the minimal contribution of dipolar coupling to the T_2^* in irradiated quartz, but is still quantifiable and confirms the proper operation of the FPGA system.

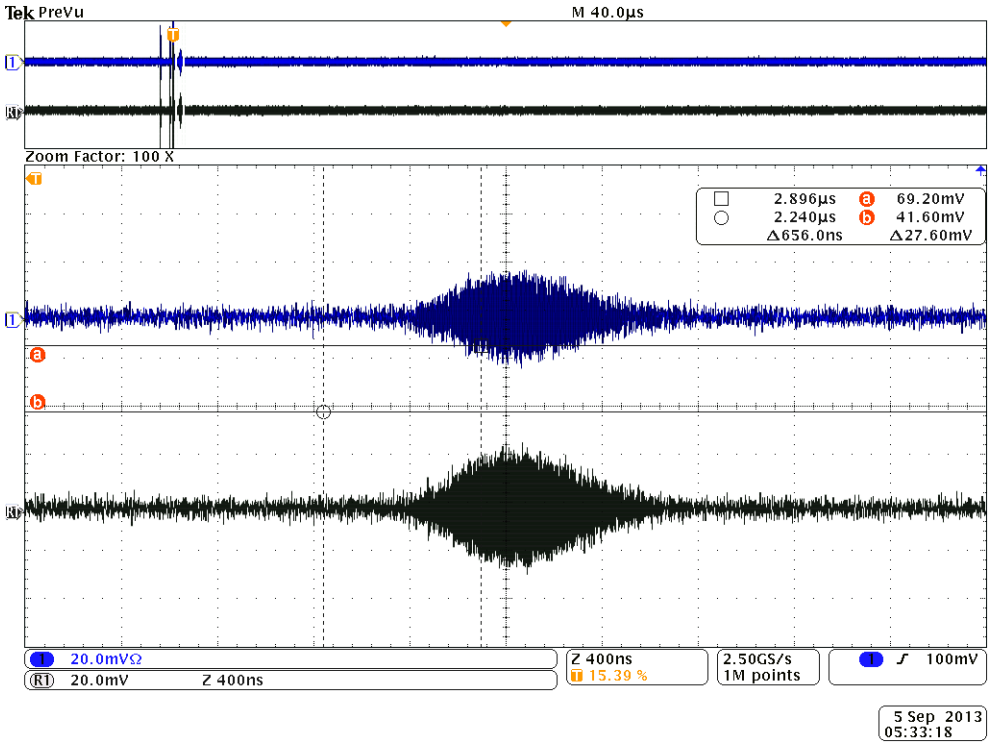

Figure 3.8: Oscilloscope trace showing spin echoes generated with identical pulse widths and τ delays. The top trace has the same-phase second pulse, while the bottom includes the $\pi/2$ phase shift. The bottom echo is of larger amplitude as expected. Here, $\tau = 4\mu s$, pulse width is 60ns, and $B_o = 3360G$.

This experiment is one that was not previously possible using the old equipment. On resonance leakage has been avoided, the setup accurately tracks the phase of the system for the pulses, and it produces the waveforms necessary for single-sideband conversion in the IQ mixer. Since the X6-1000M does not need to explicitly store the zeros in the delay between pulses, experiments requiring much longer delays (samples with a high T_2) can be easily done, while still tracking the phase, without added technical difficulty.

τ (μs)	90-90 Echo (mV)	Solid Echo (mV)
2	23	32
4	15	28
6	15	23
8	11	19
10	11	16
12	7	12
14	7	10
16	noise	7
18	noise	≈5
20	noise	noise

Table 3.1: Results of spin-echo experiment. Measurements were taken with 8 averages each. Pulses were set to 60ns wide, with a 200MHz intermediate frequency. The static field applied to the cavity was 3360G. Pulses were not calibrated to be $\pi/2$.

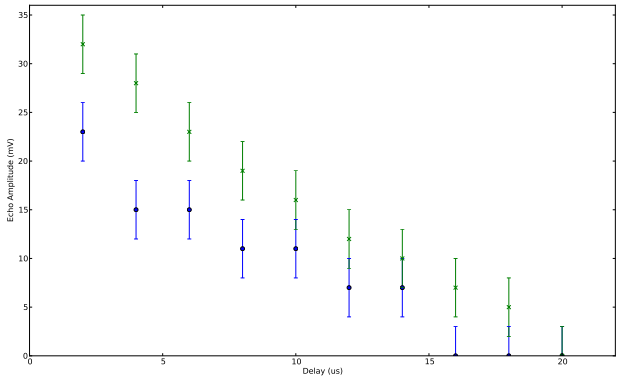

Figure 3.9: Plot of spin echo data found in table 3.1. Upper trace is the solid echo, lower is the 90-90 echo.

3.3 ODMR in Nitrogen-Vacancies

Another application for this system is use within optically detected magnetic resonance (ODMR) spectrometers. ODMR is similar to ESR in that microwave control of the spin system is used In ODMR the system response is measured by photon counting. For nitrogen-vacancy (NV) experiments, the sample is a single spin in a confocal microscope and detection is via a single-photon detector. By performing many repetitions of the same experiment enough statistics can be gathered and information about the measured state can be constructed.

To show the X6-1000M in an ODMR experiment, the board and supporting hardware was retrofitted into an existing ODMR setup being used on NVs. The FPGA was used in place of the AWG and is used to gate the RF pulse set to the power amplifier.

To show the X6-1000M used in this environment, a basic Rabi experiment was performed. The RF pulse gating time was swept from 20ns to $2\mu s$ at 20ns intervals. Each data point was repeated 50 000 times before progressing to the next. After every data point was obtained once, the NV system was automatically recalibrated to account for the sample drifting (and thus loosing focus on the NV). After this was completed, the entire scan started again from the beginning. This allows for a significant number of runs to complete while still occasionally realigning the movement stages for optimum measurements.

The results of this experiment are shown in figure 3.10 and the corresponding Fourier transform in figure 3.11. In the Fourier transform graph, there are two peaks. The first corresponds to the Rabi frequency of the NV. The second, at $\approx 14MHz$ shows that the targeted NV was coupled to a nearby Carbon atom. Coupling to a ^{13}C introduces a beating as shown in figure 3.10.

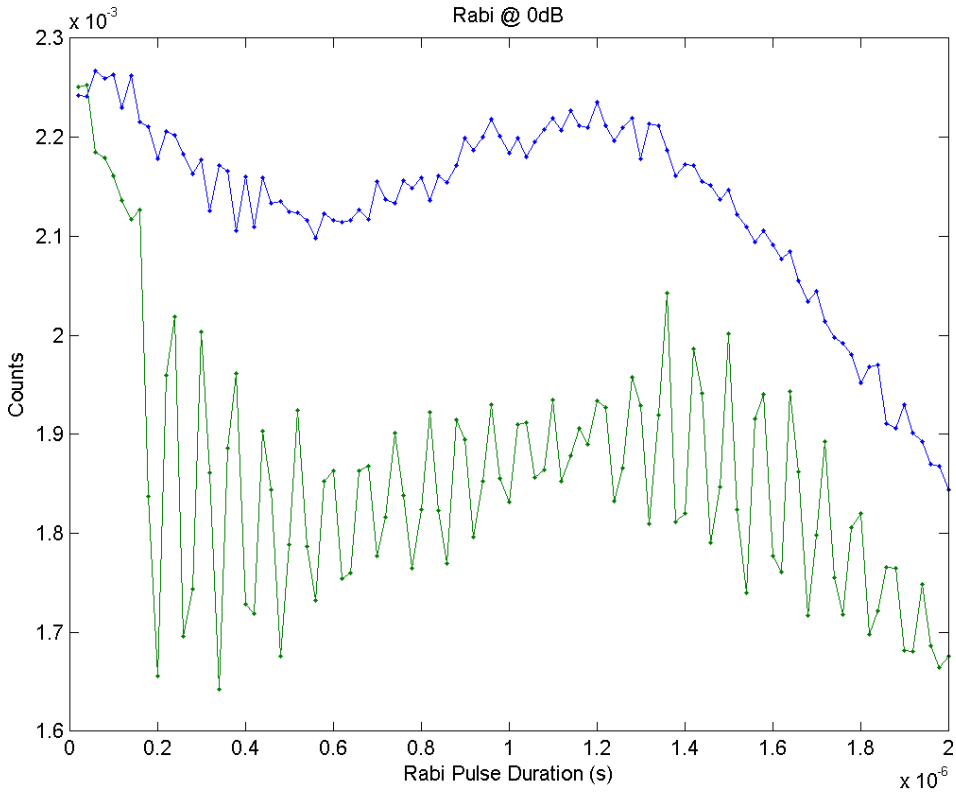

Figure 3.10: ODMR NV rabi experiment data using the the X6-1000M to gate the RF pulse.

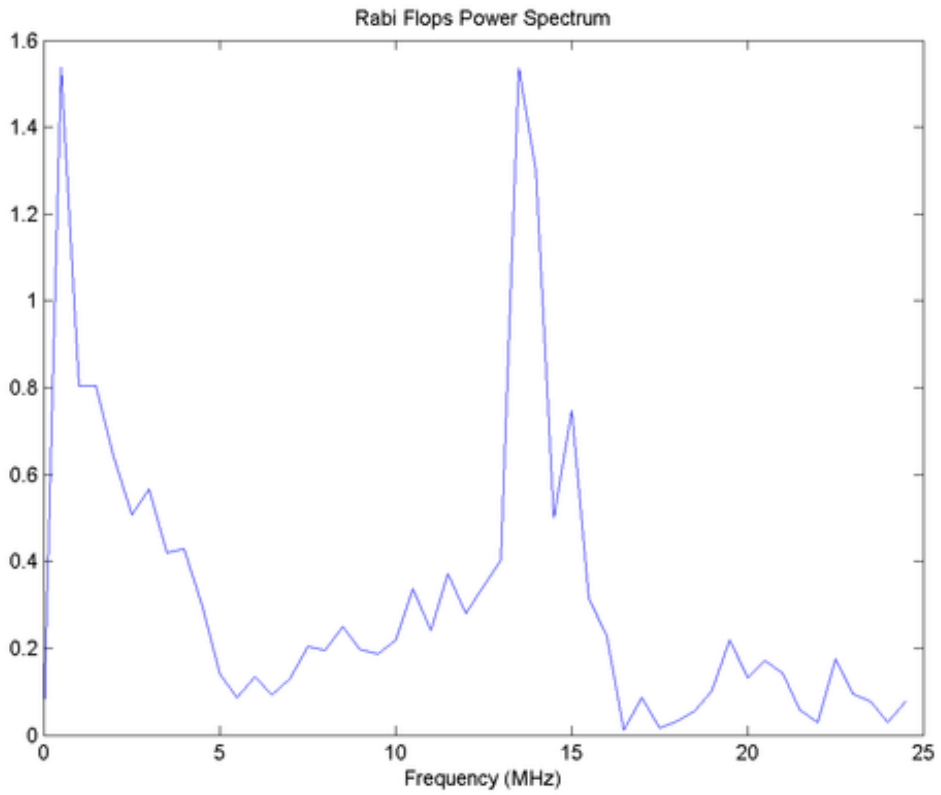

Figure 3.11: Fast-Fourier Transform of data in figure 3.10. The second peak at ≈14MHz offset indicates that the NV was coupled to a nearby Carbon spin.

Chapter 4

Conclusions

In this thesis, we have presented a new microwave spectrometer based on an FPGA development board. The new system features superior analogue performance, synchronous digital outputs, and acquisition channels all on a single compact piece of hardware. The analogue outputs run at 1GSPs, allowing for direct generation of frequencies up to 500MHz. This allows for arbitrary output amplitudes at 1ns intervals. A custom FPGA configuration allows the board to avoid using on-board memory storing zeros during pulse delays. The digial outputs can be adjusted at a 4ns resolution and are accurate down to the clock source. All of these characteristics are desirable for the core of a spectrometer.

The setup was shown to function in two different experimental setups. In the ESR setup, two experiments were performed. The first, a ringdown compensation experiment, we showed that the X6-1000M is capable of producing a superior result to that of the previous equipment and virtually eliminating the ringdown. The second was a solid echo experiment, which was not possible on the previous equipment due to lack of phase control. The other involved retrofitting the X6-1000M into an existing ODMR setup where the board was used to perform a rabi experiment on an NV. This in and of itself was capable of being done with the old equipment, however our demonstration shows that the X6-1000M can also perform existing tasks with relative ease.

Chapter 5

Next Steps

With the core of the project completed, further work can now take place to improve the system. The following is a list of actions that would make for excellent next steps.

- Hardware: Additional revisions of the DIO breakout board featuring additional space between connectors would be convient for end users. The FPGA configuration first needs to be updated to fix the occassional phase sync issues. This is currently being worked on, and the latest version greatly reduces the occurance and severity of this issue. The option to move the FPGA directly into the PC also exists and should be investigated to see if the additional communication bandwidth (≈4x speedup) this offers would be beneficial.
- Software: The PC side software has come along very nicely, but there is always more to write. Rewriting current experiment software in Python will allow for the FPGA system to be more easily used. Additionally, a calibration library would be an excellent next step, as this would allow all experiments to take advantage of a consistent software interface for calibration.
- Experiments: Expanding on the ESR experiments in chapter 3 would involve obtaining an FID from the ringdown compensation and using a different sample for the spin echos. The ODMR NV experiments have much more future work, which involves a complete rewrite of the PC side control software, further integration of the FPGA into the experimental setup, and performing pulse sequences not previously possible.

References

[1] A. Abragram. *The Principles of Nuclear Magnetism*. Oxford University Press, 1961.

[2] T. Borneman and D. Cory. Bandwidth-limited control and ringdown suppression in high-q resonators. *J. Magn. Reson.*, 225:120–129, 2012.

[3] Troy William Borneman. *Techniques for Noise Suppression and Robust Control in Spin-Based Quantum Information Processors*. PhD thesis, Massachusetts Institute Of Technology, Dec 2012.

[4] COPACOBANA. A codebreaker for des and other ciphers. http://www.copacobana.org/.

[5] Derrick Kaseman (UC Davis). Solid state experiments. http://chemwiki.ucdavis.edu/Physical_Chemistry/Spectroscopy/Magnetic_Resonance_Spectroscopies/Nuclear_Magnetic_Resonance/NMR%3A_Experimental/Solid_State_Experiments.

[6] Python Software Foundation. Python language reference, version 2.7. http://python.org/.

[7] U. Haeberlen and J.S. Waugh. Coherent averaging effects in magnetic resonance. *Physical Review*, 175(2):453–467, 1968.

[8] E.L. Hahn. Spin echoes. *Physical Review*, 80(4):580–594, 1950.

[9] J. D. Hunter. Matplotlib: A 2d graphics environment. *Computing In Science & Engineering*, 9(3):90–95, 2007.

[10] Techmag Inc. Solid-echoes. www.tecmag.com/pdf/Quadecho.pdf.

[11] Texas Instruemnts. Ads5400. http://www.ti.com/product/ads5400.

[12] Texas Instruemnts. Dac5682z. http://www.ti.com/product/dac5682z.

[13] National Instruments. What is emitter coupled logic (ecl)? http://digital.ni.com/public.nsf/allkb/BDAC410DD4A7D98B86256D040065BBD1, July 2007.

[14] Texas Instruments. Sn74lvc8t245. http://www.ti.com/product/sn74lvc8t245.

[15] Innovative Integration. einstrument - daq node. http://www.innovative-dsp.com/products.php?product=Cabled%20eInstrument.

[16] Innovative Integration. X6-1000m. http://www.innovative-dsp.com/products.php?product=X6-1000M.

[17] Dave Jones. Eevblog 496 - what is an fpga? http://www.youtube.com/watch?v=gUsHwi4M4xE, July 2013.

[18] Eric Jones, Travis Oliphant, Pearu Peterson, et al. SciPy: Open source scientific tools for Python, 2001–.

[19] KiCAD. Kicad eda software suite. http://www.kicad-pcb.org.

[20] H. Hu M.A. Smith and A.J. Shaka. Improved broadband inversion performance for nmr in liquids. *J. Magn. Reson.*, 151:269–283, 2001.

[21] NumPy. Numpy. http://www.numpy.org/.

[22] W. Hill P. Horowitz. *The Art of Electronics, second edition.* Cambridge University Press, 1989.

[23] M Bond R. Clayton. Experience using a low-cost fpga design to crack des keys. http://www.cl.cam.ac.uk/~rnc1/descrack/DEScracker.html.

[24] NXP Semiconductor. Pbss2515vs. http://www.nxp.com/documents/data_sheet/PBSS2515VS.pdf.

[25] ON Semiconductor. *A Comparison of LVDS, CMOS, and ECL.* Application Note AND8059/D. April 2001.

[26] Wikipedia. Emitter-coupled logic. http://en.wikipedia.org/wiki/Emitter-coupled_logic.

[27] Wikipedia. Field-programmable gate array. http://en.wikipedia.org/wiki/Field-programmable_gate_array.

[28] Wikipedia. Gratis versus libre. http://en.wikipedia.org/wiki/Gratis_versus_libre.

[29] Wikipedia. Kicad. http://en.wikipedia.org/wiki/KiCad.

[30] Wikipedia. Protocol of bitcoin - bitcoin mining. http://en.wikipedia.org/wiki/Protocol_of_Bitcoin#Bitcoin_mining.

[31] Xilinx. Virtex-6 fpga family. http://www.xilinx.com/products/silicon-devices/fpga/virtex-6/.

APPENDICES

Appendix A

DIO Breakout Board Images

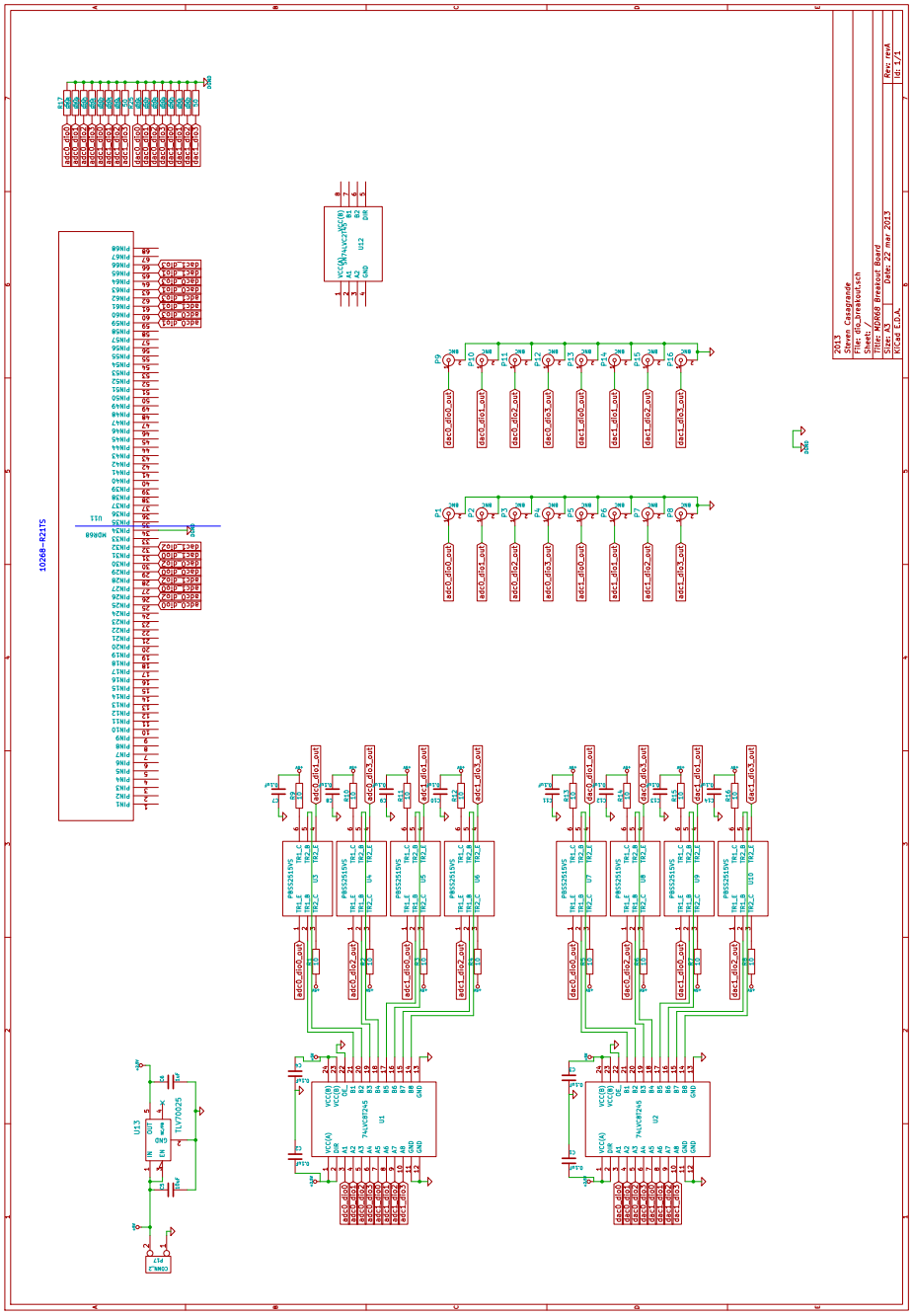

Figure A.1: Schematic for X6-1000M DIO breakout board made using KiCAD.

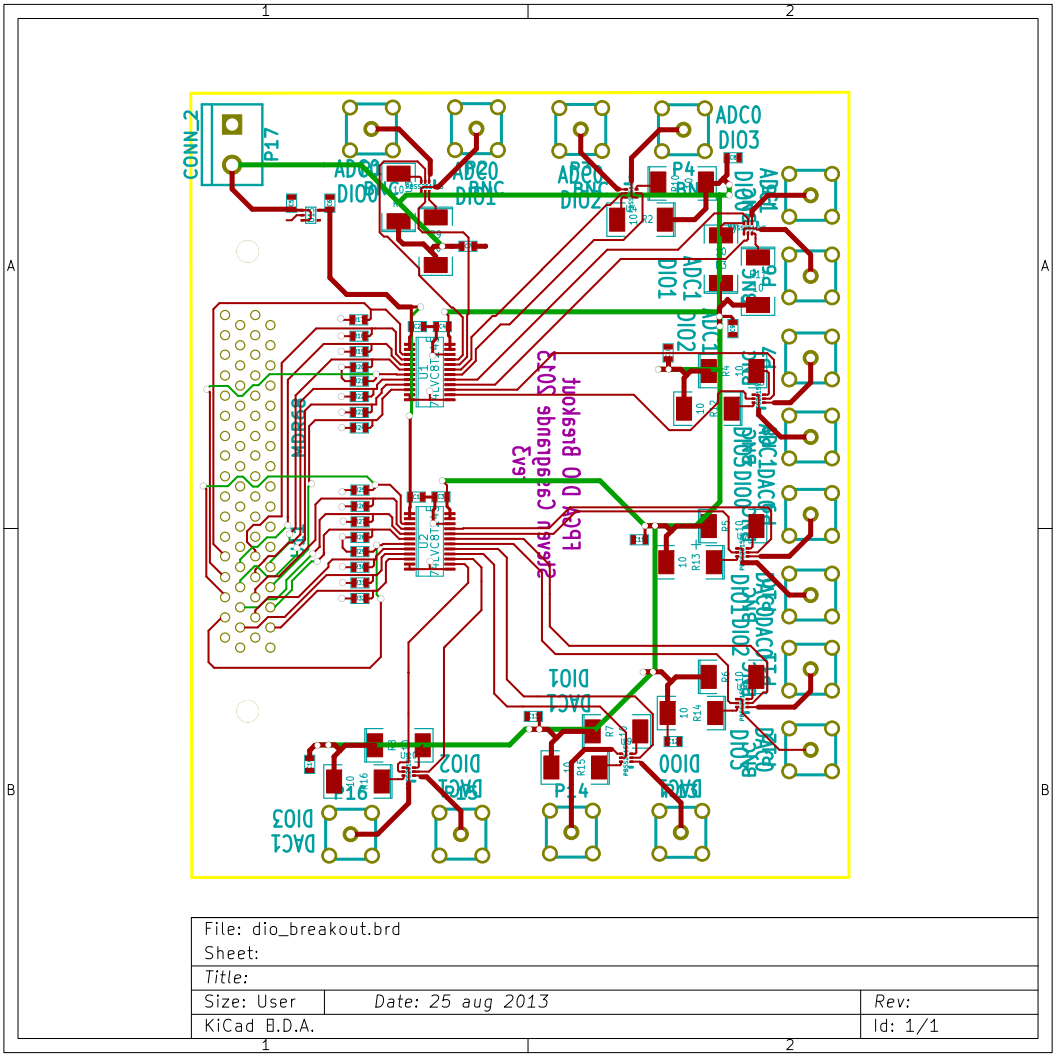

Figure A.2: PCB layout for X6-1000M DIO breakout board made using KiCAD. This diagram omits the filled copper zones on the front and back for ease of viewing.

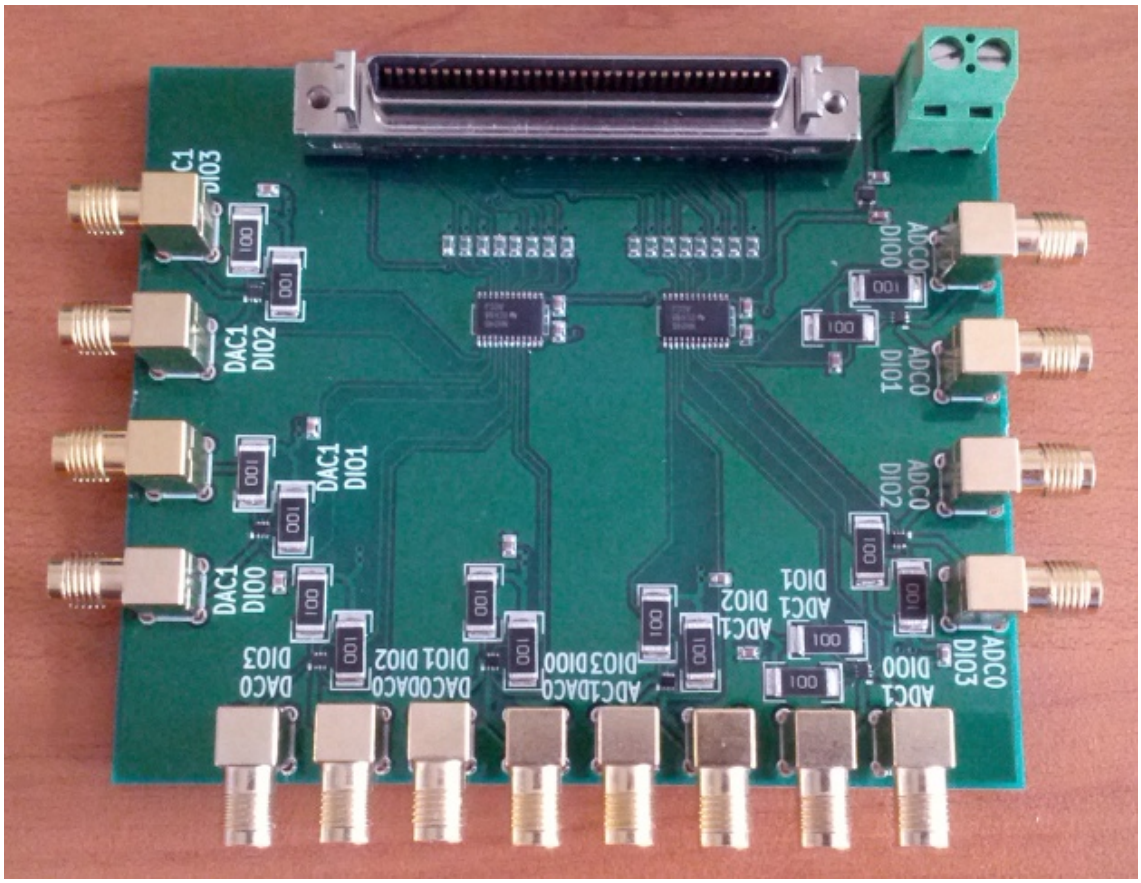

Figure A.3: Photograph of of revision 3 of the DIO breakout board. The large connector seen at the top is the primary MDR68 connector. The screw-terminal is the power input. All other connectors are SMA. PCB manufacture is APCircuits. Board assembly was performed by myself.

Appendix B

X6-1000M Images

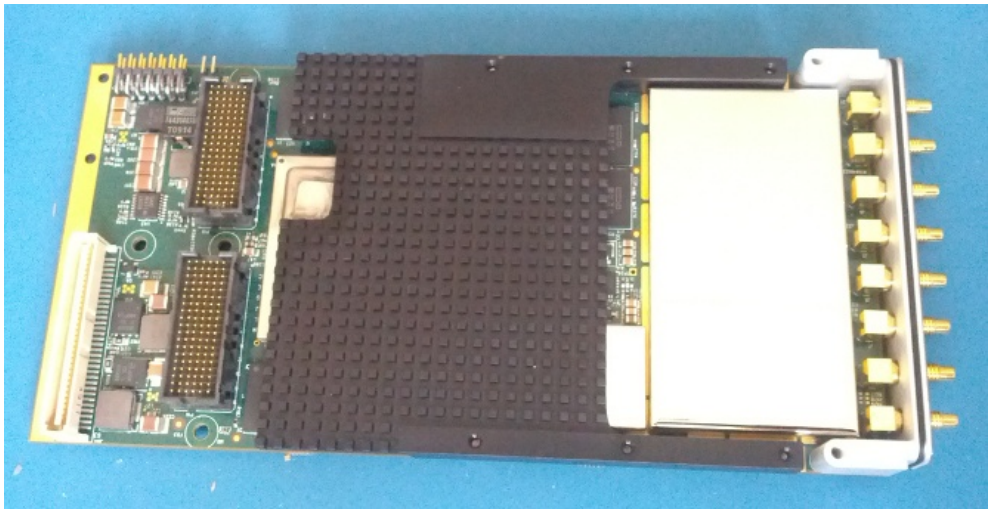

Figure B.1: Photograph of the X6-1000M board[16]. The FPGA is located under the heatsink. All the analogue circuitry is located under the shielding on the right. Connectors on the left are used for exposing all the board's communication lines to the enclosure (see figure B.2)

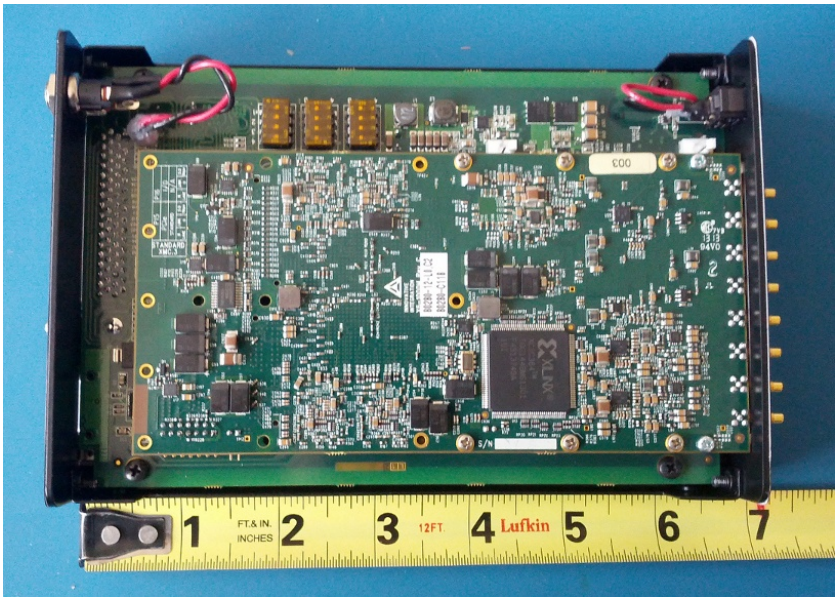

Figure B.2: Photograph of the X6-1000M board[16] plugged into the eInstrument[15] enclosure which is used to connect to the board to a PC. Cover is missing to show some of the internals.

Figure B.3: Photograph of the X6-1000M board[16] plugged into the eInstrument[15]. Front connectors shown here are the SSMC variety.