A Fully Optical and Scalable Network with All-to-All Programmable Connections

James Williams

August 2019

1 Abstract

Optical networks present a fundamentally new approach to computation and simulation. Previous partially-optical networks have been demonstrated which can solve the Ising problem with all-to-all connections. These networks are limited by the electronics required to simulate the optical interactions. Combined with a novel network design, a new generation of electronics enables an all-optical, scalable network configuration which circumvents problems encountered in the old design. This architecture can be adapted to solve the Ising problem as well as to simulate other physical models. We present the aforementioned network along with the cross-domain FPGA used to implement the high bandwidth signals which control the interactions within the network.

2 Introduction and Background

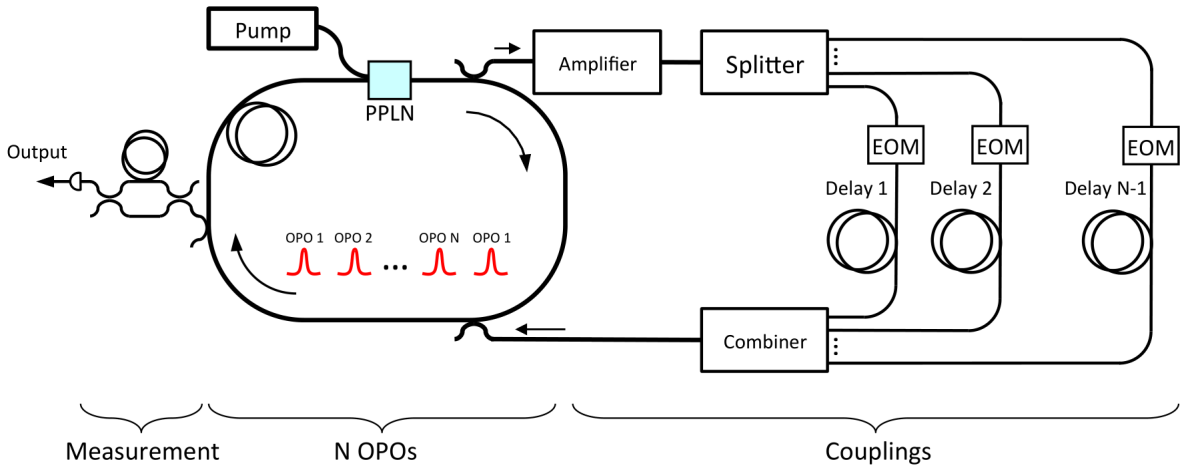

Figure 1: A schematic of the fiber-based scalable optical network found in (1). The electrical control system developed during the 10 week period will drive the electro-optical modulators (EOM) in the right of the figure.

A primary focus of the Marandi research group is the development of an all-optical network architecture that can solve a variety of nondeterministic polynomial (NP) hard problems. NP-hard problems are ubiquitous in the modern world and appear frequently in many fields. Examples such as the traveling salesman problem (2), the scheduling problem (3), and the Ising problem (4) are encountered on a daily basis and require both fast and accurate solution methods. Solutions to these problems are crucial for finding, for example, optimal shipping routes, the most efficient order of task execution, and for exploring new areas of physics. Our group has successfully implemented a partially optical Ising machine that can find the ground state of systems with up to 100 spins and

10,000 spin-to-spin connections (4). The machine works by using a series of optical pulses to represent each spin in the system. The "up" and "down" states of each spin are mapped to the 0 and π phase states of each optical pulse. The interactions matrix J, which represents each spin's interaction with every other spin, is implemented by using the FPGA to apply feedback to each pulse.

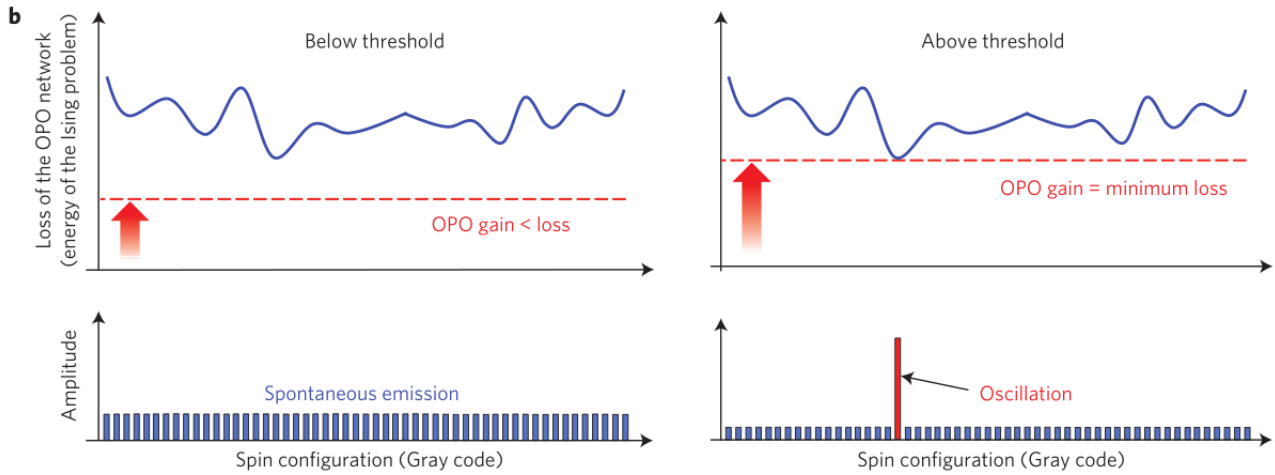

Figure 2: A diagram of the Hamiltonian search process taken from (1).

To solve the Ising problem, the energy space of the Hamiltonian (H) is searched by pumping the system at an energy below the oscillation threshold of the optical parametric oscillator (OPO), and then increasing the pumping energy until oscillation is observed (see Fig. 2). During this pumping phase, the probability amplitudes of ground states are amplified through constructive interference while the probability amplitudes of all other states are suppressed through destructive interference. Once oscillation begins, all non-ground states are completely suppressed, and the system continues to oscillate in one of the degenerate ground states. The performance of this machine is limited by the FPGA used to simulate the interactions between spins. We plan on using the all-optical version of this machine to study topological photonics (5) by implementing the Harper-Hofstadter model in an optical network where each optical pulse would represent a lattice site. The long term goals of this project are the development of a general, scalable optical network architecture that can be modified to fit any NP-hard problem, and development of the fast electrical drivers needed to implement on-chip optical networks.

3 The Problem

Because an all-optical Ising machine using our optical setup requires modulation rates greater than 250 MHz, a new waveform generator must be developed to support higher frequencies, as typical arbitrary waveform generators have maximum bandwidths of around 100 MHz and lack the required number of channels. The requirements for the new waveform generator are as follows:

- Must have a bandwidth greater than 250 MHz.
- Must have at least 16 channels.
- Each channel must be independently programmable with an arbitrary waveform.
- Must interface with current LabView setup.
- Channels must be phase-locked to the experiment reference clock.

In order to meet these requirements, the ZCU1275 development board from Xilinx (6) was chosen. The ZCU1275 offers 16 DAC channels, each with a bandwith of 4 GHz. Each channel is independently accessible over an AXI stream interface so that they can be programmed to playback an arbitrary waveform. The reference clock from each DAC can be set to an arbitrary value, which allows for synchronization and phase locking with the experiment reference clock. The development board also offers a USB to UART interface to allow for a simple messaging system to be implemented between a PC running LabView and the board.

While this development board will enable the implementation of an all-optical Ising machine, it will also enable larger problems to be solved. The increase in frequency allows more pulses to be modulated and maintained within the optical cavity, leading to a simulation consisting of more spins with more interactions. This advantage will also move us closer to our long-term goal of implementing a chip-scale version of our tabletop optical network.

4 Methods: The FPGA

4.1 Features

4.1.1 Waveform Playback

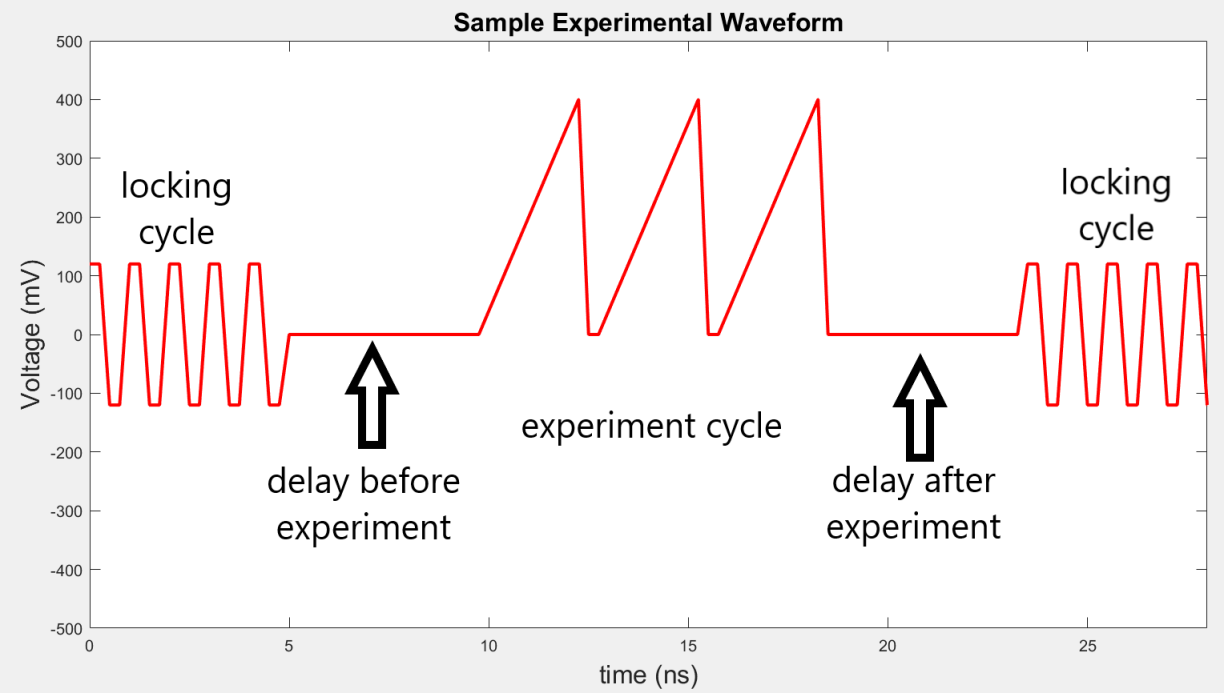

Figure 3: A sample experimental waveform played back by the FPGA for driving the electro-optic modulators present in the optical cavity.

The main advantage of the FPGA is its ability to output an arbitrary waveform provided by the user which conforms to specific timing requirements imposed by the experiment. The FPGA has a sample rate of 4 GSPS and hence a sample resolution of 250 ps.

Before the experiment begins, the FPGA allows a “locking cycle” waveform to play back continuously, permitting a small amount of light to enter the cavity. This light allows the electronics responsible for monitoring the length of the network to “lock” the network to a specified length. The locking cycle waveform returns automatically after the conclusion of the experiment cycle and post-experiment delay.

Once a trigger signal is received, the locking cycle ends and a delay before the experiment begins. This delay is configurable in steps of 250 ps and allows the light from the locking cycle to decay before the experimental light is injected into the cavity. This delay is also crucial for accommodating the various lengths of both the coaxial cables responsible for carrying signals from the FPGA to the modulators and the varying lengths of fiber optic cable within the network. The user can also create a delay after the experiment has ended in order to leave enough time for data to be read out before the locking cycle begins again.

The experimental waveform provided by the user can contain up to 65536 samples and can be repeated an arbitrary number of times. This allows multiple experimental runs to be undertaken in one trigger, which enables experimental data to be averaged in order to improve the signal-to-noise ratio (SNR). Experimental waveforms need only to be provided by the user in the form of a CSV file. All processing between the CSV file and the data needed by the FPGA to successfully play back the waveform is undertaken by the provided PC drivers discussed in section 4.4.1.

4.1.2 Waveform Capture

Along with the digital-to-analog converters (DAC) offers by the FPGA, 16 analog-to-digital converters (ADC) are also available for use. These ADCs are used by the FPGA to capture and process experimental data. The ADCs sample at 2 GSPS and allow for up to 524288 samples to be captured. The FPGA hardware design also implements an averaging feature for improved SNR. The PC driver allows the user to control all capture settings, and to save and view the data captured by the FPGA. Sanity checks are also implemented which prevent the user from offloading the FPGA data prematurely before enough samples have been taken for a complete averaging cycle.

4.2 Hardware Description

4.2.1 Top Level Design

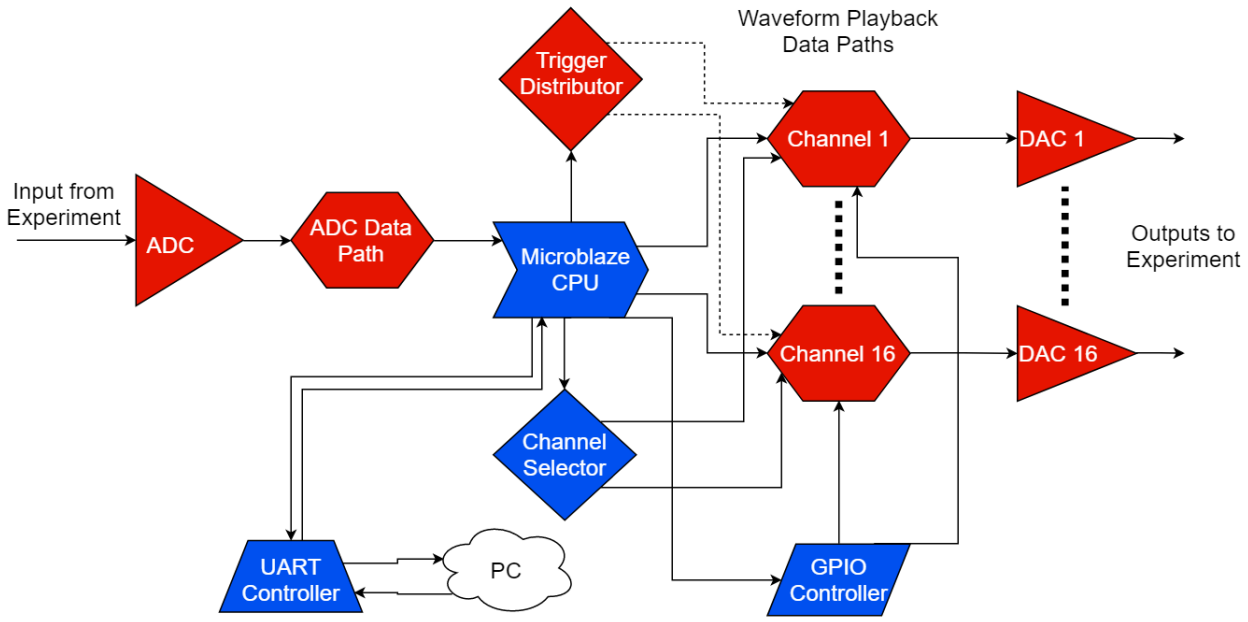

Figure 4: A block diagram of the top level design of the FPGA firmware.

The top-level design consists of three main components: the DAC logic, the ADC logic and the Microblaze CPU. The CPU sits at the center of the design and controls all DAC and ADC functions along with the GPIOs and UART connections. The CPU has 16 AXI stream outputs, each of which is used by the 16 DAC channels. These

outputs first go to the waveform playback data pathway, and from there to the RF data converter IP from Xilinx. This IP abstracts away all RF functionality of the FPGA. The CPU also contains 16 AXI stream inputs. 4 of these are used to read data captured from the ADC by the waveform capture data pathway.

The top level design also includes two blocks used for managing the DAC datapaths: the trigger distribution module and the channel selector. The trigger distribution module was added to solve a timing issue. It was noticed during testing that some channels would trigger 4 ns early or late with respect to channel 1, and that this skew would change between triggers. The problem was solved by assigning one register to each trigger signal and having these registers send a trigger pulse within the RF clock domain to ensure all DACs started on the same clock cycle. The channel selector works in conjunction with the GPIO controller to select and upload settings to a particular channel.

4.2.2 Waveform Playback Hardware

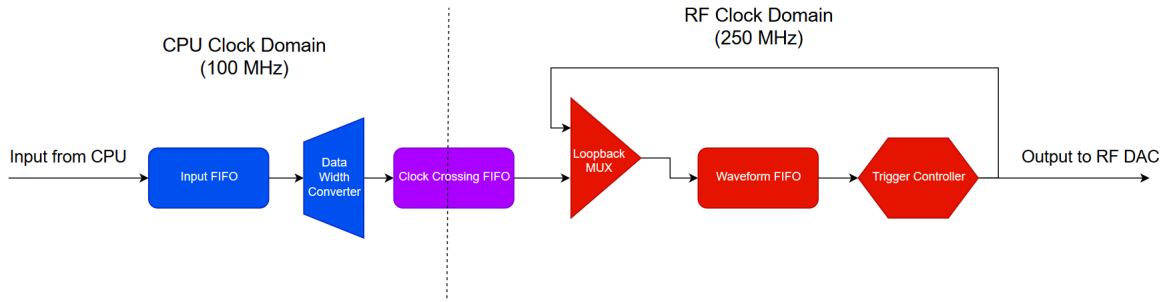

Figure 5: A diagram of the waveform playback data path used by each channel. Blocks in blue belong to the CPU clock domain while blocks in red belong to the RF clock domain. The purple block allows data to cross between the clock domains.

Each DAC requires sets of 16 16-bit 2's complement samples (a single DAC word) to be provided at a rate of 250 MHz. Because this is much faster than the on-board CPU which provides the waveforms for the DACs, a buffering stage is required. This buffer stage is used to read the waveform from the CPU at a relatively slow rate and then write the waveform to the DAC at a higher rate.

The data path (fig.5) performs three tasks. First, it allows a waveform to be streamed from the CPU into a local high-speed first-in first-out (FIFO) buffer. Second, it allows that buffer to play back the waveform to the DAC when a trigger is received from either the CPU or an external source. Third, it allows the waveform to be played back to the DACs a user-defined number of times before flushing the waveform from the buffer. The data path also includes several shift registers which control the delay before the experiment waveform playback, and the locking cycle waveform.

The first two tasks are implemented using a series of FIFO buffers along with some custom logic. After the waveform data leaves the CPU, it enters a data width converter which combines eight 32-bit words from the CPU into one 256-bit word for the DAC. This data stream is sent to a second FIFO, the clock crossing FIFO, which enables the data to cross over from the CPU clock domain to the DAC clock domain. The stream is then stored inside of a much larger (4096 words deep) FIFO, the waveform FIFO, in preparation for playback on the DAC. Between the output of this larger FIFO and the input of the DAC is a small piece of custom logic, the trigger controller, which allows the CPU to interface with the stream control signals. This ensures that the CPU can disable this stream so that the waveform can be stored in the FIFO, and later enable it so that the waveform can be played back over the DAC.

The third task is implemented by using a hardware counter inside of the trigger controller. After the waveform is loaded into the FIFO, the trigger controller is loaded with a 32-bit count value via a shift register, indicating the number of clock cycles during which the waveform should be played back. The output of the trigger controller is also connected to the input of a MUX between the clock crossing FIFO and the waveform FIFO which

allows the waveform to be looped back into the FIFO to enable repeated playback. Once the trigger signal is received, the trigger controller allows the data stream from the FIFO to the DAC to run for the prescribed amount of clock cycles before resetting itself for the next trigger pulse. The trigger controller also provides other control inputs for clearing the FIFO of any remaining waveforms and overriding the internal counter to trigger continuously.

The locking cycle waveform is set by loading a 16 sample waveform into an internal shift register via the channel selector and GPIO controller. This waveform is played back indefinitely until the start of the experiment. Playback resumes once the experiment has ended. The delay before the experimental waveform defines a period of zero signal output from the DAC between the end of the locking waveform and the beginning of the user provided waveform. This delay can be adjusted in steps of 250 ps. In order to achieve this resolution, a pre-waveform buffer, implemented as a shift register, was used as an intermediary so that the waveform can be shifted on a per sample basis (instead of delaying in steps of one full 4 ns clock cycle). For delays longer than 4 ns, another shift register is used to count clock cycles before the one cycle playback of the pre-waveform. The data for these shift registers is automatically generated and uploaded by the PC driver on channel initialization.

4.2.3 Waveform Capture Hardware

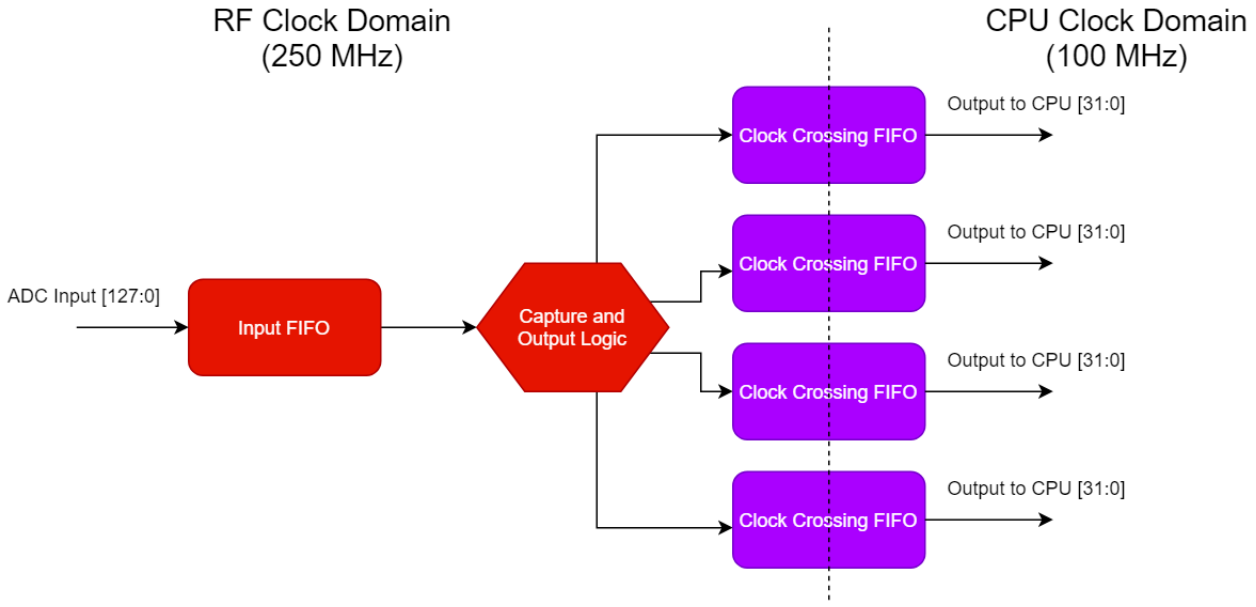

Figure 6: A diagram of the data path used to move data from the ADC to the CPU. Blocks in red are in the RF clock domain while blocks in purple are used to cross clock domains.

The on-board ADC has been enabled in order to record and store experimental data without the need for an oscilloscope. A custom IP was implemented which allows data to be received from the ADC using a capture trigger signal provided by the CPU. This data can then be uploaded and viewed on the user's PC using the python driver. The ADC samples at 2 GSPS and can sample a maximum of approximately 262 μs. The "capture and output logic" (Fig. 6) also implements an averaging feature, allowing the user to average concurrent captures to reduce the amount of noise present in the signal. The averaging works by using a simple bitshift to implement a 2^n hardware divide. While averaging, the ADC simply acts as an accumulator to add and store incoming samples to the previously accumulated samples.

Because the ADC operates much faster than the CPU, a buffer is needed to store the waveform so that the CPU can read it out at a later time. The purple FIFO buffers in Figure 6 are used to store and split up the waveform into

smaller (from 128 bits to 32) bus sizes so that the waveform can be read out using the stream interfaces provided by the CPU. These four separate streams are recombined in software to display the waveform.

4.3 CPU Firmware

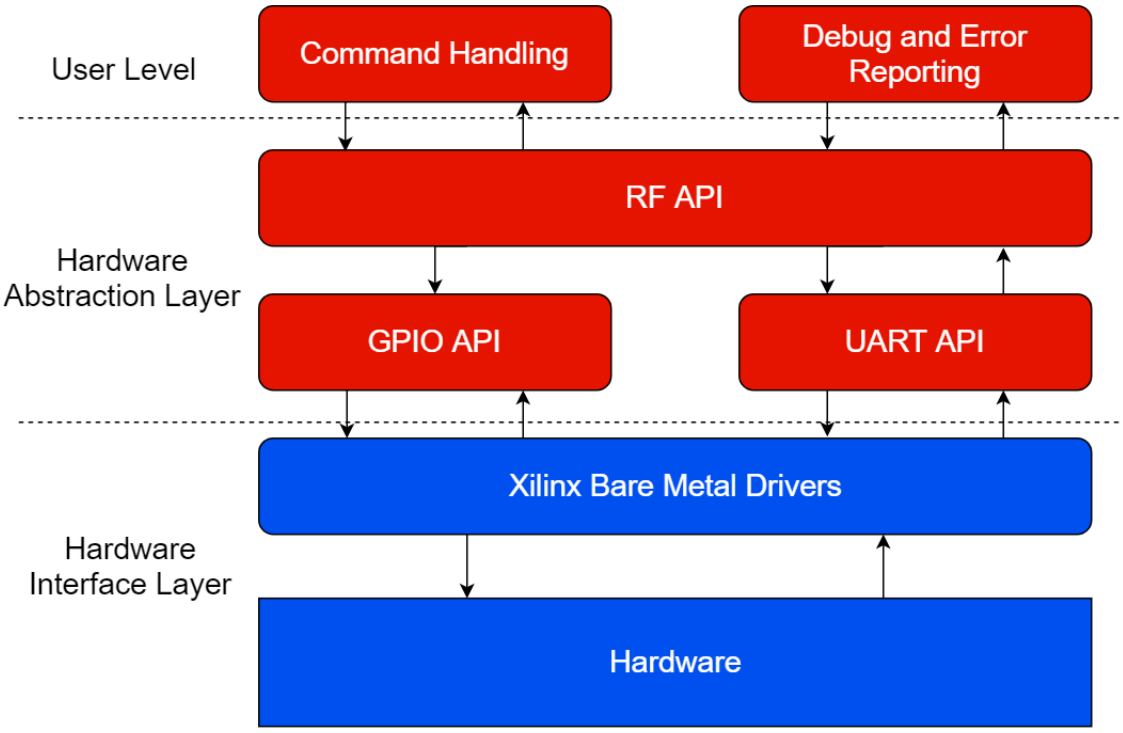

Figure 7: A diagram of the firmware used to drive the Microblaze CPU.

The Microblaze CPU (Fig. 4) uses custom firmware to implement the configuration and control functionality of the FPGA via a UART interface. The CPU firmware allows the user to check the status of all hardware components, configure waveform playback properties on a per-channel basis, configure the ADC capture time and averaging, trigger remotely and read out the ADC data.

At the top level of the firmware stack (Fig. 7) are the user-level functions. These implement the command processing via UART so that commands can be sent to and executed on the board via the PC. Several debug and error functions are also implemented to allow the PC software to check the status of the board and ensure that all clock connections are operational. Checking to ensure that hardware clocks are running is crucial as the CPU may inadvertently halt itself and require a reset if it attempts to read or write to hardware which is not receiving a clock.

In the middle of the firmware stack are the two separate abstraction layers which implement high-level functions such as flushing the waveform buffers, loading waveforms, setting channel configurations, and initializing the hardware on boot. This layer was implemented in order to provide a more intuitive user application programming interface (API) as the Xilinx drivers lack detailed documentation and can cause a system halt if used incorrectly.

4.4 PC Control and Integration into Experiment

4.4.1 Python Drivers and Labview Control

In order to control the experiment from a nearby PC, a UART-based driver was written in Python to implement full control over the board without needing to send individual UART commands. This driver facilitates a control system

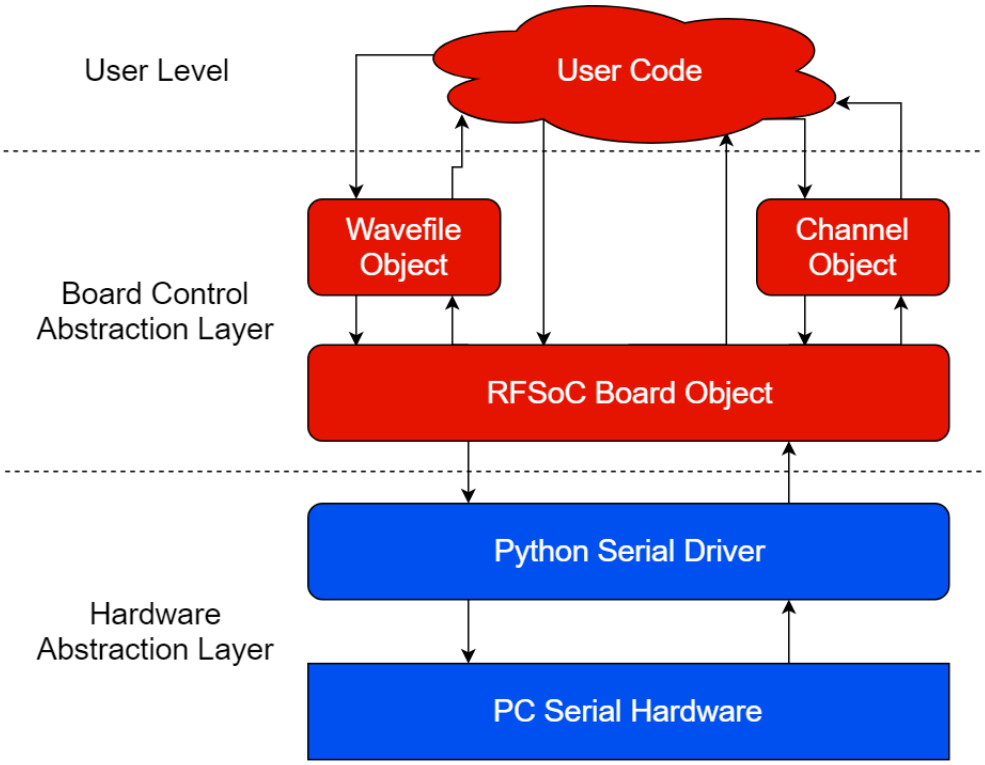

Figure 8: A diagram of the PC driver used to implement LabVIEW control.

between the board and a LabVIEW front panel. The front panel works by calling individual Python scripts with a list of arguments to implement board initialization, channel configuration, remote triggering, and data collection. The Python driver loads and saves the state of the board on every function call, allowing LabVIEW to call each function in an independent terminal instead of preserving a Python environment across function calls.

The Python driver implements a set of three objects which allow the user to configure the properties of the experimental and locking cycle waveforms, configure the channel settings, and control the board. The Wavefile object (fig.8) implements the byte stream encoding needed by the RF DACs. The user supplies a comma separated value file (CSV) containing the waveform data as an argument of the Wavefile constructor. The constructor then interpolates the waveform and translates it to a series of 16-bit 2's complement values needed by the DAC. It then performs any necessary shifts or amplitude scaling requested by the user. The Channel object takes two Wavefiles, an experiment Wavefile and a locking cycle Wavefile, as arguments, and allows the RFSoC Board object to access various bytestreams for upload to the board such as the delay values, locking cycle bytestream, and experiment waveform bytestream.

The RFSoC Board object implements the majority of the functions needed to check the status of and control the board. It has several safeguards build in to ensure that the user does not inadvertently crash the board by specifying an incorrect value. This object also allows the user to plot the waveforms he or she has provided for playback, and to view the waveforms captured by the ADC during the experiment.

4.4.2 Connections to Electro-optic Modulators

In order to control the optical pulses within the network, electro-optic modulators must be used to interface the FPGA with the network. These modulators translate an electronic RF signal into a delay modulation or amplitude modulation for the pulses within the network. One crucial aspect of controlling these modulators is aligning the electronic signals with the optical pulses so that they arrive at the modulator simultaneously. Previously, this was

difficult, as differing cable lengths between the FPGA and the modulators necessitated phase shifts with a resolution of 500 ps. With the advent of the faster hardware provided by the FPGA, the phase shift of each signal can be adjusted in steps of 250 ps, allowing each signal to be adjusted to various cable lengths and amplifier delays.

5 Testing and Results

5.1 Tevelopment Board Testing

The development board was first tested using sample waveforms and a 20 GHz 80 GSPS oscilliscope (7). The sample waveforms were compared to those captured by the oscilloscope to ensure that the waveforms were not corrupted in the process of converting them into an appropriate sample stream for the DACs. This testing uncovered several timing issues related to signals crossing from the CPU clock domain to the FPGA clock domain.

While testing waveforms designed for simulating the SSH model (8) on the optical network, several problems involving repeated waveforms were discovered. Because of the implementation of the 250 ps delay resolution, between 0 and 3.75 ns of the waveform could be shifted to the end of the waveform so that the waveform would be repeated correctly. This lead to unwanted pulses appearing at the end of the waveform which were initially present at the beginning. This was resolved through a software fix which automatically padded the end of the waveform with 0 V values when necessary.

After confirming that the development board was able to accurately playback sample waveforms and experimental waveforms, the oscilloscope was exchanged for optical phase (9) and intensity (10) modulators to confirm that the DAC outputs could modulate laser pulses appropriately. Measurements using a 12 GHz optical detector (11) confirmed that the DAC was able to produce a 0 to 2π phase shift using the phase modulator, and a -3 to -40dB intensity change using the intensity modulator.

5.2 Optical Network Testing

Due to several problems encountered while setting up the optical network, usable results have yet to be collected. A large thermal drift in the laboratory ($\pm 1^{\circ}C$) has caused problems when attempting to lock the network to a constant length. Problems with balancing the network gain and loss have also increased the difficulty in improving the number of round-trips completed by each pulse. A large number of round trips is desirable as it increases the time over which the pulses, and hence the calculation, can be observed to evolve. We plan to publish significant experimental results by the end of the year.

6 Acknowledgements

This project was sponsored in part by Southern California Edison. I would like to thank them for contributing to my research. I would also like to thank Professor Marandi for the opportunity to work in his group this summer.

References

[1] A. Marandi, Z. Wang, K. Takata, R. L. Byer, and Y. Yamamoto, "Network of time-multiplexed optical parametric oscillators as a coherent ising machine," *Nature Photonics*, vol. 8, no. 12, p. 937, 2014.

[2] J. K. Lenstra and A. R. Kan, "Some simple applications of the travelling salesman problem," *Journal of the Operational Research Society*, vol. 26, no. 4, pp. 717–733, 1975.

[3] R. Ruiz and J. A. Vázquez-Rodríguez, "The hybrid flow shop scheduling problem," *European journal of operational research*, vol. 205, no. 1, pp. 1–18, 2010.

[4] P. L. McMahon, A. Marandi, Y. Haribara, R. Hamerly, C. Langrock, S. Tamate, T. Inagaki, H. Takesue, S. Utsunomiya, K. Aihara *et al.*, "A fully programmable 100-spin coherent ising machine with all-to-all connections," *Science*, vol. 354, no. 6312, pp. 614–617, 2016.

[5] L. Lu, J. D. Joannopoulos, and M. Soljačić, "Topological photonics," *Nature Photonics*, vol. 8, no. 11, p. 821, 2014.

[6] "Zynq ultrascale rfsoc zcu1275 characterization kit." [Online]. Available: https://www.xilinx.com/products/boards-and-kits/zcu1275.html

[7]

[8] H. Takayama, Y. R. Lin-Liu, and K. Maki, "Continuum model for solitons in polyacetylene," *Physical Review B*, vol. 21, no. 6, p. 2388, 1980.

[9] "ixblue mpz-ln-10 phase modulator." [Online]. Available: https://photonics.ixblue.com/product/md/mpz-ln-10

[10] "ixblue mx-ln-10 intensity modulator." [Online]. Available: https://photonics.ixblue.com/product/md/mx-ln-10

[11] "Fiber-optic receiver, nir, 12 ghz, 500-1630 nm, fc singlemode." [Online]. Available: https://www.newport.com/p/1544-A