1 23 4 5 6 7

Development of ROACH firmware for microwave multiplexed X-ray TES microcalorimeters

T. J. Madden, *Member, IEEE,* T. W. Cecil, L. M. Gades, O. Quaranta, D. Yan (闫代康), A. Miceli, D.T. Becker, D.A. Bennett, J.P. Hays-Wehle, G.C. Hilton, J.D. Gard, J.A.B. Mates, C.D. Reintsema, D.R. Schmidt, D.S. Swetz, L.R. Vale, J.N. Ullom

***Abstract*—We are developing room temperature electronics based upon the ROACH platform to readout microwave multiplexed X-ray TES. ROACH is an open-source hardware and software platform featuring a large Xilinx Field Programmable Gate Array (FPGA), Power PC processor, several 10 GB Ethernet SFP+ interfaces, and a collection of daughter boards for analog signal generation and acquisition. The combination of a ROACH board, ADC/DAC conversion daughter boards, and hardware for RF mixing allows for the generation and capture of multiple RF tones for reading out microwave multiplexed X-ray TES microcalorimeters. The FPGA is used to generate multiple tones in base band, from 10 MHz to 250 MHz, which are subsequently mixed to RF in the multiple GHz range and sent through the microwave multiplexer. The tones are generated in the FPGA by storing a large lookup table in Quad Data Rate (QDR) SRAM modules and playing out the waveform to a DAC board. Once the signal has been modulated to RF, passed through the microwave multiplexer, and has been modulated back to base band, the signal is digitized by an ADC board. The tones are modulated to 0 Hz by using a FPGA circuit consisting of a polyphase filter bank, several Xilinx FFT blocks, Xilinx CORDIC blocks (for converting to magnitude and phase), and special phase accumulator circuit for mixing to exactly 0Hz. Upwards of 256 channels can be simultaneously captured and written into a bank of 256 First-In-First-Out (FIFO) memories, with each FIFO corresponding to a channel. Individual channel data can be further processed in the FPGA before being streamed through a 10 GB Ethernet fiber-optic interface to a Linux system. The Linux system runs software written in Python and QT C++ for controlling the ROACH system, capturing data, and processing data.**

***Index Terms*—FPGA, Firmware, ROACH, microwave multiplexing, TES, superconducting microcalorimeters, superconducting detectors**

I. Introduction

A DATA acquisition system for reading microwave multiplexed Transition Edge Sensors (TES) has been developed based on the ROACH platform [1]. The system consists of a 100 mK cryostat housing NIST-designed Transition Edge Sensors coupled to Superconducting Quantum Interface Devices (SQUID), which in turn are coupled to microwave resonators [2]. The ROACH system, consisting of large Xilinx FPGA board, an ADC/DAC board, called the MUSIC board, and an intermediate frequency (IF) mixing board simply called the "IF Board" [3]. The FPGA generates signals sourced by DACs from 10 MHz to 250 MHz, that are mixed to 5 GHz to stimulate the microwave resonators. After passing through the cryostat, the signal is mixed down to base band on the IF Board, and digitized by ADCs to be analyzed by the FPGA. When X-rays impact on the TES, the resulting electrical pulse alters the phase and amplitude of the signal going through the microwave resonator. The ROACH FPGA analyzes this signal and extracts the X-ray pulse. A diagram of the system is in Figure 1. The ROACH system is based on IQ (In-phase and Quadrature) modulation, meaning that two DACs and two ADCs are required for signal generation and digitization. The goal of this work is to develop firmware which will not degrade the energy resolution of hundreds of microwave multiplexed X-ray TES readout simultaneously. In this paper, we present our progress towards this goal. In particular, we present details on the firmware specifically designed to readout microwave multiplexed X-ray TES.

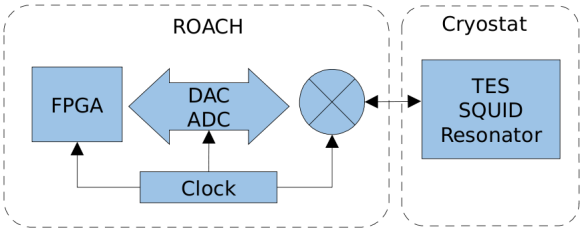

Fig. 1. ROACH TES Data Acquisition System. ROACH system includes FPGA, ADC/DAC conversion, and IQ mixing. The FPGA/DAC generates base band signals between 10 MHz and 250 MHz that are mixed to 5 GHz before being sent to the cryostat, that contains TES sensors. Signals returning from cryostat are mixed down to below 250 MHz, digitized, and analyzed by the FPGA.

II. Firmware Design

The firmware was designed at Argonne National Laboratory for the ROACH II hardware, hereafter called ROACH. The ROACH board features a Xilinx Virtex 6 and accompanying Power PC processor providing a Linux software interface to

This research is supported by the Accelerator and Detector R&D program in Basic Energy Sciences Scientific User Facilities (SUF) Division at the Department of Energy. This research used resources of the Advanced Photon Source and Center for Nanoscale Materials, a U.S. Department of Energy Office of Science User Facilities operated for the DOE Office of Science by Argonne National Laboratory under Contract No. DE-AC02-06CH11357.

Corresponding Author: T. Madden is with Argonne National Laboratory, Chicago, USA e-mail: tmadden@anl.gov.

T. Cecil, L. Gades, A. Miceli, and O. Quaranta are with Argonne National Laboratory

D. Yan is with Northwestern University and Argonne National Laboratory

D. Bennett, J. Hays-Wehle, G. Hilton, C. Reintsema, D. Schmidt, D. Swetz, L. Vale are with National Institute of Standards and Technology

J. Gard, J. Mates are with University of Colorado, Boulder

D. Becker, and J. Ullom are with National Institute of Standards and Technology and University of Colorado, Boulder

registers in the Virtex 6. The tool flow is the Casper MATLAB-Simulink front end to Xilinx System Generator, where most firmware is written by drawing MATLAB block diagrams. Software was written in Python for control of the ROACH and in C++ for data capture over 10 GB Ethernet. Our firmware design was originally based upon ROACH I firmware written by the Mazin group at UCSB, [4] which was designed to readout optical Microwave Kinetic Inductance Detectors and thus requires significant modifications to readout X-ray TES using microwave SQUID multiplexers. Our current firmware currently supports ROACH II, and returns data specifically useful for TES detectors. Below we describe the firmware design and we point out the modifications and improvements relative to the UCSB firmware, and specific to our design.

An important consideration when designing FPGA firmware is the clock rate at which the firmware must run. The DAC/ADC hardware runs at a sample rate of 512 MHz. Because FPGAs generally cannot run at this frequency, the FPGA must either run at half the ADC rate at 256 MHz, or one quarter of the ADC rate at 128 MHz. Running the FPGA at a higher speed can create timing violations due to placement and routing. Running the FPGA at a slower speed requires more logic and DSP resources to accommodate four simultaneous streams rather than two. This consideration has far reaching effects on every firmware block in the design.

After having trouble meeting timing constraints working with the two-stream UCSB firmware, we chose to redesign the firmware based on four streams, thus halving the FPGA clock speed. Ramifications of this design choice include a forced redesign of the wave form generator, design of a four-port ADC/DAC interface in Verilog for ROACH, and redesign of the final mixing to 0 Hz. Further, some of the four-port blocks in the Casper library had bugs, which motivated a redesign of the FFT using Xilinx FFT blocks. Another advantage to running the FPGA at 128 MHz rather than 256 MHz is that the DRAM on ROACH cannot operate fast enough for a 256 MHz design. After compiling and debugging our design, we found that we had ample FPGA resources despite the four-tap design. Our design currently uses only 22% of available slices, 15% of Block RAM (BRAM), and 6% of DSP48's.

The design of our firmware consists of two major parts. The first part generates a plurality of complex sinusoid tones and feeds them to two DACs for the real (I) and imaginary signals (Q). Because the DACs run at 512 MHz and the FPGA is clocked at 128 MHz, four samples per DAC are generated on a single FPGA clock cycle and sent to the DACs.

The second part of the design consists of a dual ADC interface that captures complex I and Q data after being sourced through the cryostat. Because the ADCs run at 512 MHz, four samples per ADC are captured and processed by the FPGA per clock cycle. The samples are windowed with a hamming window using a ROACH library Polyphase block before being sent to an FFT [5]. The windowed data is processed by a 512-point complex FFT. The FFT coefficients, representing "channels" with one TES per channel, are then converted from rectangular to polar coordinates with a Xilinx CORDIC block before being stored in a bank of FIFOs, with one FIFO per channel. These FIFOs are addressable, with the address selecting the FIFO or channel of interest. By storing into FIFOs each channel can be operated upon as a continuous stream independent of other channels. The firmware supports up to 256 channels at once. Because the source frequency may not be at exactly an FFT bin center frequency, a phase correction circuit operates on the phase data to effectively mix the signal to 0 Hz (relative to the bin center frequency). Following the phase correction circuit is a flux ramp demodulation circuit. Information on what the flux ramp demodulation circuit is for, and how it works can be found in [6].

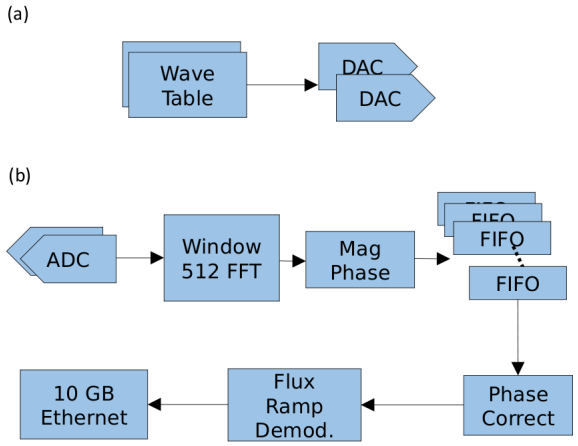

Fig. 2. Firmware Design. (a) QDR resident on the ROACH board stores sinusoids of many frequencies for an I and Q signal. Two DACs are used to generate the I and Q signals. (b) Two ADCs capture I and Q signals retrieved from the cryostat. The signals are processed with a complex FFT, converted to magnitude and phase, and arranged into individual streams with a FIFO bank. Each channel stream is mixed to exactly 0 Hz with phase correction, Flux Ramp Demodulated, and sent to a Linux system via 10 GB Ethernet.

The data is finally sent to a local network via 10 GB Ethernet using the fiber optic link included on the ROACH board. Data can be streamed in various formats, including raw noise data, noise plus flux ramp demodulation, or only flux ramp demodulated data. A Linux computer system captures the streamed data for further analysis. A diagram of our firmware design is in Figure 2. Each channel has a sample rate of 1 MHz.

III. Details of Firmware Blocks

A. FFT Block

Although the Casper Library contains an FFT block that is used in the UCSB firmware, we found that when computing four samples per clock there were problems with compiling the block. For this reason, a new block was designed using Xilinx FFT blocks [7]. Because the Xilinx FFT can only process one sample per clock (i.e., only accepts a single input data stream) four Xilinx FFT blocks were instantiated in the design, with each FFT block accepting its own input data stream. To compute a 512 point FFT, four 128-point FFT blocks are used. The outputs of four FFT blocks, with each producing complex outputs from complex inputs, are combined with "butterflies," or complex multiply-accumulate circuits. Coefficients for the butterflies, called "twiddle" factors, are stored in BRAM.

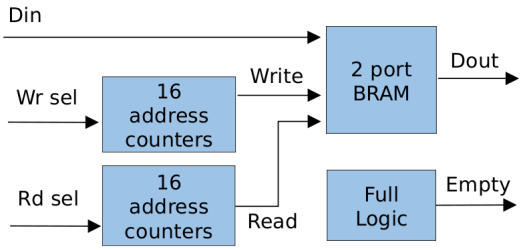

Fig. 3. FIFO bank implemented with bank of counters and two-port BRAM. FIFO is selected by channel number. Separate select lines for read and write. Larger banks created through multiplexing.

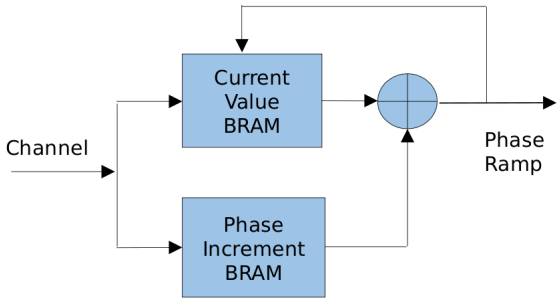

Fig. 4. Phase Correction circuit implemented as one adder that integrates a ramp. Ramp slope and current value are stored in BRAMs and indexed by channel number.

The resulting circuit is a 512 point FFT that accepts four simultaneous data streams, and produces four output data streams. The connection of FFT blocks with butterflies is based on the definition of the FFT [8].

B. Addressable FIFO Bank

Because the FFT produces data in bin order, and it is desired to operate upon a single channel or bin, the bins of interest are stored to FIFOs, with a FIFO assigned to each bin. The FIFO is selected with an address or channel both for reading and for writing. Reading and writing can be done on different channels simultaneously. Because of large FPGA resources required to implement the FIFO bank with actual Xilinx FIFO blocks, the FIFO bank was implemented with a BRAM block and two banks of address counters for reading and writing. One BRAM is associated with 16 address counters to essentially create 16 FIFOs. To implement 256 FIFOs, FIFO banks are multiplexed.

The firmware uses a BRAM and state machine to map FFT bin indices to channels, that define which FIFO stores a particular stream of data. See Figure 3 that shows the FIFO bank design. The UCSB firmware relied on a "Commutator" circuit for channelization, that is functionally equivalent to a FIFO Bank. However a FIFO Bank allows a more modular design based on "producers" and "consumers" that are separate firmware blocks.

C. DAC/ADC Interface

When the firmware was designed there existed no DAC or ADC converter four-tap interface for the MUSIC DAC board for ROACH [3]. A new "yellow block," a ROACH community term for certain Simulink blocks in the Casper library, was created in Verilog to interface to the MUSIC DAC/ADC board. This design was based on the ROACH I yellow block written by Bruno Serfass, Sean McHugh and Ran Duan. [3].

D. Phase Correction Circuit

When capturing sinusoids from the cryostat generally the sinusoid frequency does not fall exactly in the FFT bin center. The result is that the FFT mixes the signal to near to but not exactly 0 Hz. For bin k at frequency ω_k the FFT coefficient X_{ω_k} is computed as

$$X_{\omega_k} = \sum_{n=0}^{N-1} h_n e^{i\omega_s n} e^{-i\omega_k n} = h_n * e^{i(\omega_s - \omega_k)n}, \quad (1)$$

which is the convolution of the window function h_n with the frequency error between bin center frequency ω_k and signal frequency ω_s. The FFT is a block transform in that it operates upon 512 samples at some clock time, then operates upon the next 512 samples exactly 512 sample clocks later. For signals not at bin center, the phase for each successive in time

$$\angle X_{\omega_k} = (\omega_s - \omega_k)512m, \quad (2)$$

where m is a time index denoting successive FFT computations, equal to 512 ADC sample clocks and 128 FPGA clocks. It is apparent that the phase term is an ever increasing ramp. To provide useful data, this ramp is removed by subtracting a ramp of same slope. The circuit is implemented as a single adder with one BRAM storing the current ramp value for 256 channels, and a second BRAM storing the phase increment or ramp slope for 256 channels. The phase naturally wraps every 2π radians. Figure 4 shows the Phase Correction circuit design. The Phase Corrector circuit, designed specifically for our firmware and not appearing in the UCSB firmware, was designed because the four-tap data flow of our firmware used up more external DRAM then the 2-tap data flow. The Phase Corrector replaces many MB of DRAM usage with a few kB of FPGA-resident BRAM.

E. Wave Generation

The signal to be sent to the cryostat is a summation of complex sinusoids, or cosines sent to the "I" DAC, and negative sines sent to the "Q" DAC. The original UCSB firmware written for ROACH I used DDR. However, we found the DDR hardware on our ROACH II system to be unstable. We chose to use the two Quad Data Rate (QDR) static RAMs resident on the ROACH board to store the tones. Because the Casper library provides a software addressable interface to the QDR's the waveforms can be easily sent to the QDR with a Python script. Reading out the QDR's and sending the signal to the DAC's is done with a state machine written with a Xilinx M-Code block. Because the firmware operates on four samples per clock, the QDR must store a total of eight samples per address, or four 16-bit values for the I and Q signals. Because this requires a data width of 128 bits, two QDR blocks are needed in the design.

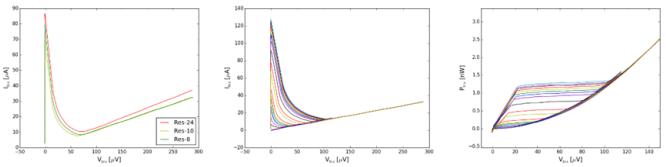

Fig. 5. IV curves obtained with ROACH. (Left) The current-voltage (IV) curves of three microwave multiplexed hard X-ray TES taken simultaneously using the ROACH. This demonstrates the ability of the ROACH to read out multiple TES simultaneously. (Center) Current-voltage (IV) curves for a single TES at different bath temperatures taken using the ROACH. (Right) The IV curves plotted as power-voltage curves. The TES transition region appears as the flat regions in the curve.

F. Flux Ramp Demodulation

Flux Ramp Demodulation (FDR) is the computation of a Discrete Fourier Transform on a set of phase values of the FFT coefficients. We have implemented and tested an FDR circuit in our firmware. The design is based on [6].

G. 10 GB Ethernet Interface

The data to be sent to the local network is organized into UDP packets and transmitted via a 10 GB fiber optic link. The data sent is formatted to include headers and "packets" of streamed FFT coefficients. Because the formatted data packets do not line up with the UDP packet size, the firmware must break up data packets before sending. Software on a Linux system receiving data must reassemble the formatted data into individual channel streams. In short, the firmware breaks up the FFT coefficients into many streams for individual channel processing, then serializes the channels back into one stream only to be deserialized by software. The GB Ethernet interface is specific to our firmware because the original ROACH I, and hence UCSB firmware, had no 10 GB Ethernet interface.

IV. Software Design

No firmware is of any use without some sort of software control. Our ROACH system software has two components. First, Python scripts with Graphical User Interface(GUI) function to set up the ROACH and control it. The Python scripts borrow from the Casper library and were originally adapted from the scripts from [4]. Second, a QT C++ program was created from scratch to capture data from the 10 GB Ethernet interface. Because of the high data rates, this program runs multiple threads for capturing UDP packets (without packet loss), parsing packets into individual channels, user interface, communication with the Python scripts, detection of pulses, and data saving to disk.

V. Current Status and Future Work

Currently, we have validated our ROACH firmware's ability to measure three microwave SQUID multiplexed TES IV curves simultaneously which are shown in Figure 5. We have validated the operation of 256 channels in RF loopback mode, though optimization is needed to assure no FIFOs overflow when all channels are simultaneously read out. Because of speed limits in the 10 GB Ethernet link, we can reliably read continuous noise from 16 channels at once. Optimization of the firmware to improve the speed of the 10 GB link is planned. Future work includes implementing firmware pulse detection as well as validating the noise performance for simultaneous TES readout. The software will be improved by adding EPICS support and real-time pulse analysis [9]. Our ROACH firmware can be found online at [10].

VI. Acknowledgement

The authors would like to thank B. Mazin and his team, and Ran Duan for help on their firmware.

References

[1] Casper collaboration website. [Online]. Available: https://casper.berkeley.edu/wiki/ROACH

[2] D. A. Bennett, J. A. Mates, J. D. Gard, A. S. Hoover, M. W. Rabin, C. D. Reintsema, D. R. Schmidt, L. R. Vale, and J. N. Ullom, "Integration of tes microcalorimeters with microwave squid multiplexed readout," *IEEE Transactions on Applied Superconductivity*, vol. 25, no. 3, pp. 1–5, 2015.

[3] Music readout. [Online]. Available: https://casper.berkeley.edu/wiki/MUSIC_Readout_(Kinetic_Inductance_Detector_(KIDs))

[4] R. Duan, S. McHugh, B. Serfass, B. A. Mazin, A. Merrill, S. R. Golwala, T. P. Downes, N. G. Czakon, P. K. Day, J. Gao *et al.*, "An open-source readout for mkids," in *SPIE Astronomical Telescopes+ Instrumentation*. International Society for Optics and Photonics, 2010, pp. 77 411V–77 411V.

[5] The polyphase filter bank technique. [Online]. Available: https://casper.berkeley.edu/wiki/The_Polyphase_Filter_Bank_Technique

[6] J. Mates, K. Irwin, L. Vale, G. Hilton, J. Gao, and K. Lehnert, "Flux-ramp modulation for squid multiplexing," *Journal of Low Temperature Physics*, vol. 167, no. 5-6, pp. 707–712, 2012.

[7] Xilinx fast fourier transform. [Online]. Available: http://www.xilinx.com/products/intellectual-property/fft.html

[8] W. T. Cochran, J. W. Cooley, D. L. Favin, H. D. Helms, R. A. Kaenel, W. W. Lang, G. Maling, D. E. Nelson, C. M. Rader, and P. D. Welch, "What is the fast fourier transform?" *Proceedings of the IEEE*, vol. 55, no. 10, pp. 1664–1674, 1967.

[9] Experimental physics and industrial control system. [Online]. Available: http://www.aps.anl.gov/epics/

[10] Argonne-developed firmware. [Online]. Available: https://github.com/argonnexraydetector/RoachFirmPy