Revision of a Python GUI to Synchronize Two AWGs, Experimental Fabrication of AlN MEMS Resonators, and Analysis of Coulomb Blockade in a Single-Electron Transistor Coupled to a Single-Electron Trap

Richard McManus
Electrical Engineering
University of Notre Dame
Notre Dame, IN
rmcmanu2@nd.edu

***Abstract* – This report describes the progress made during the Fall 2023 semester in Dr. Snider's research group and focuses specifically on my contributions. The projects described consist of the revision of a Python graphical user interface (GUI) to control two Zurich Instruments arbitrary waveform generators (AWG), the experimental fabrication of aluminum nitride piezoelectric microelectromechanical systems (MEMS) resonators for implementation with an adiabatic microprocessor without interlocked pipelined stages (MIPS) to achieve Bennet clocking, and the analysis of coulomb blockade diamonds generated by a single-electron transistor (SET) coupled to a single-electron box (SEB).**

I. INTRODUCTION

This semester I had the opportunity to continue working with Dr. Snider's research group, contributing to a variety of projects. In the following sections, I will discuss the methods and processes for these contributions and the results obtained.

II. Revision of a Python GUI to Synchronize Two AWGs

A. Background

To accommodate thermal constraints, the speeds of modern microprocessors are restricted to well below their RC time constants. While existing methods such as dark silicon set out to combat the breakdown of Denard scaling, the pursuit of viable alternatives, such as adiabatic computing [1], capable of reducing active power persists. Our group designed an adiabatic MIPS to meet this need and is preparing to test its performance. To perform computations adiabatically, the MIPS requires Bennet Clocking [2]. This project was intended to generate 12 trapezoidal waveforms to provide Bennet Clocking to the adiabatic MIPS during testing. Following generation, these 12 signals will be inverted by external circuitry on a printed circuit board (PCB) to generate 24 clock signals: 12 positive clock signals and 12 negative clock signals. A block diagram of this process is shown in Fig. 1.

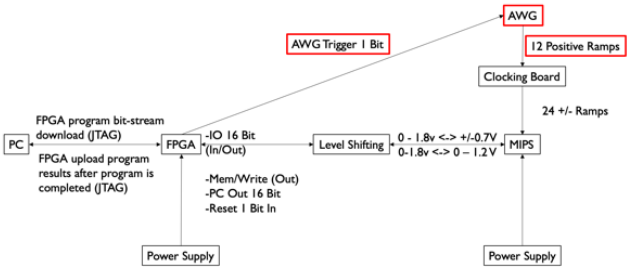

Fig. 1. Block diagram of MIPs testing. The AWG is intended to provide 12 positive trapezoidal waveforms to the clocking board. The clocking board will generate 12 additional inverted waveforms to create a total of 24 bipolar nested trapezoidal waveforms.

This project builds upon three years of preparation for the testing of our adiabatic MIPS. In previous semesters, waveforms for Bennett Clocking were generated with hardcoded values by cutting and joining rectangular and ramp waveforms [3]. This method is shown in Fig. 2.

```
//Waveform 1 (2t,1t,4t,1t,2t) (zeros,ramp,rect,rampdown,zeros)
wave w_flat = rect(4*samples, 1.0); // Generates rectangle wave with constant
wave w_trap1 = join(w_zero, w_rise, w_flat, w_fall,w_zero); // combines them

//Waveform 2 (2.5t,1t,3t,1t,2.5t) (zeros,ramp,rect,rampdown,zeros)
wave w_flat2 = rect(3*samples, 1.0); // Generates rectangle wave with constan
wave w_trap2 = join(w_zero2,w_rise, w_flat2, w_fall,w_zero2); // combines the

//Waveform 3 (3t,1t,2t,1t,3t) (zeros,ramp,rect,rampdown,zeros)
wave w_flat3 = rect(2*samples, 1.0); // Generates rectangle wave with constan
wave w_trap3 = join(w_zero3,w_rise, w_flat3, w_fall,w_zero3); // combines the

//Waveform 4 (3.5t,1t,1t,1t,3.5t) (zeros,ramp,rect,rampdown,zeros)
wave w_flat4 = rect(640, 1.0); // Generates rectangle wave with constant 1
wave w_trap4 = join(w_zero4,w_rise, w_flat4, w_fall,w_zero4); // combines the
```

Fig. 2. Initial method of trapezoidal waveform generation. Waveforms are created by cutting and joining rectangular and ramp waves.

During the Fall semester of 2021, a Python script [4] was developed to generate waveforms from a comma-separated values (CSV) file and output these waveforms on up to eight channels of a single AWG. This script ran from the command line and was incapable of adjusting attributes such as frequency and sample rate. An example output of trapezoidal waveforms generated by this Python script is shown in Fig. 3.

Fig. 3. Trapezoidal waveforms generated by a Python script ran from the command line. The waveforms were programmed using CSV files generated through Origin.

Following this development, the synchronization of multiple AWGs was desired. In the Spring Semester of 2023, development began to create a graphical user interface (GUI) to automate the process of AWG synchronization and waveform generation. It was desired for the GUI to input waveforms from a CSV file and allow users to specify attributes such as frequency, sample rate, and triggers before generation. Such a GUI would enable custom use cases and expedite the process of programming.

During this development, a bug was discovered in Zurich Instruments' Lab One application programming interface (API) multi-device kernel (MDK). As Bennet clocking was critical to the success of the project, an alternative method of achieving AWG synchronization was required. Through the implementation of sequence and sample clock delays and waveform augmentation, a Python GUI was produced that automated the synchronization and programming process [5]. This GUI enabled the synchronization of two 8-channel AWGs within 200 picoseconds as shown in Fig. 4.

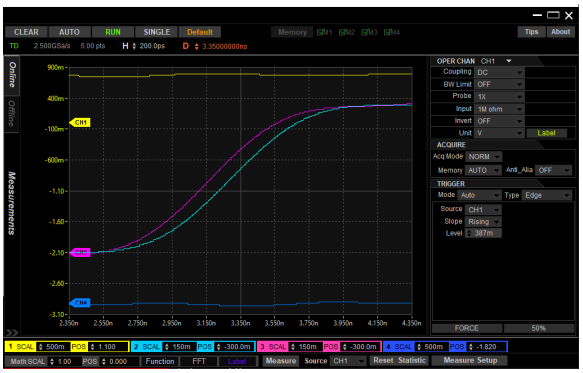

Fig. 4. Synchronization of two 8-Channel AWGs using clock delays. A delay of less than 200 ps is shown between the two devices.

This semester, the previously developed Python GUI was revised through the implementation of Zurich Instruments's intended method of device synchronization. Through collaboration with the Zurich Instruments Support Team, the group was provided access to a beta version of the Zurich LabOne API with a patch for the MDK bug. This beta version was utilized to develop an updated Python GUI to automate synchronization and waveform generation.

B. Implementation of Zurich Multi-Device Synchronization

Zurich Instrument's intended method for multi-device synchronization (MDS) uses dedicated MDS trigger ports in a loop and requires an external 10 or 100-MHz reference clock. A Rigol DG4162 was selected to provide the external 10-MHz reference clock signal. The required configuration for this method of synchronization is shown in Fig. 5.

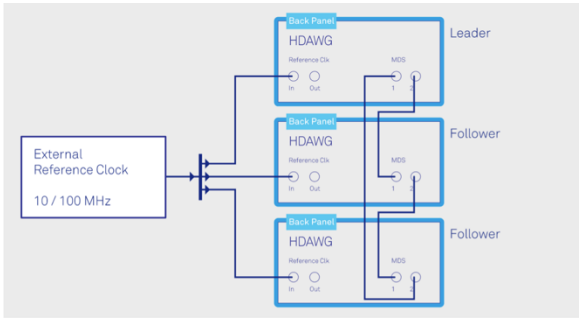

Fig. 5. Cable configuration of AWGs for MDS. An external 10 or 100-MHz clock is required as an input to all devices. MDS trigger ports should be configured in a loop.

While a Python script, mds.py, was created in a previous semester containing a variety of user-defined functions to execute the multi-device synchronization protocol, numerous revisions were required to implement the updated MDK and achieve successful synchronization.

C. Results of Python GUI Implementing MDS

The Python GUI developed during this semester to implement Zurich Instrument's MDS functionality is shown in Fig. 6 and 7. Details regarding the GUI's functionality and instructions for use were provided in Richard McManus's Spring 2023 report [4].

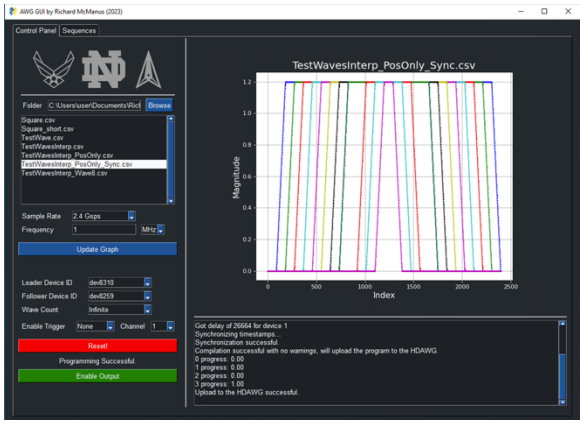

Fig. 6. Control panel of Python GUI implementing Zurich Instrument's MDS.

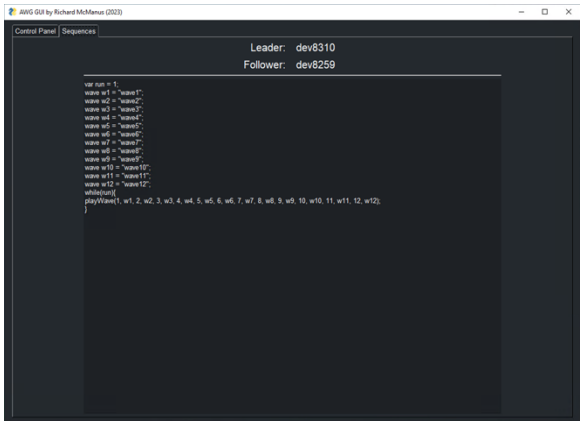

Fig. 7. Sequence tab of Python GUI implementing Zurich Instrument's MDS.

The implementation of Zurich Instrument's MDS functionality yielded less desirable performance than the previously developed GUI utilizing clock delays. As shown in Fig. 8, the MDS functionality achieved a delay of over 500 ps between devices. This delay is nearly triple that achieved by the previous workaround method shown in Fig. 4.

Fig. 8. Synchronization of two 8-Channel AWGs using Zurich Instrument's MDS functionality. A delay greater than 500 ps is shown between the two devices.

D. Conclusions and Next Steps

As the synchronization achieved using MDS was less accurate than that of the previously developed method, the group is required to decide which method to implement. While the utilization of clock delays achieves higher performance, it suffers from requiring manual tuning following every AWG power cycle. Alternatively, the implementation of Zurich Instrument's MDS functionality does not require manual tuning. However, this method requires an external reference clock and achieves a larger synchronization delay of more than 500 ps.

Should the group decide to utilize Zurich Instrument's intended MDS functionality, further work will be required to initiate the AWGs from the GUI itself. Currently, to achieve waveform generation, the the Python GUI requires AWG initiation through Zurich Instruments's LabOne user interface. Collaboration with Zurich's support team will be required to implement this functionality.

III. Experimental Fabrication of AlN MEMS Resonators

A. Background

While AWGs will be utilized to provide Bennet Clocking during testing, a method of generating ramping clock signals on-chip is required for the widespread adoption of adiabatic reversible logic. This semester, the group fabricated AlN MEMS resonators to evaluate their potential for integration with an adiabatic MIPS to provide Bennet Clocking. This project builds upon the work of Michael S. McConnell [6] and Rene Celis Cordova [7] to develop piezoelectric contour mode resonators (CMRs). One circuit utilizing CMRs to generate nested ramp waves is shown in Fig 9.

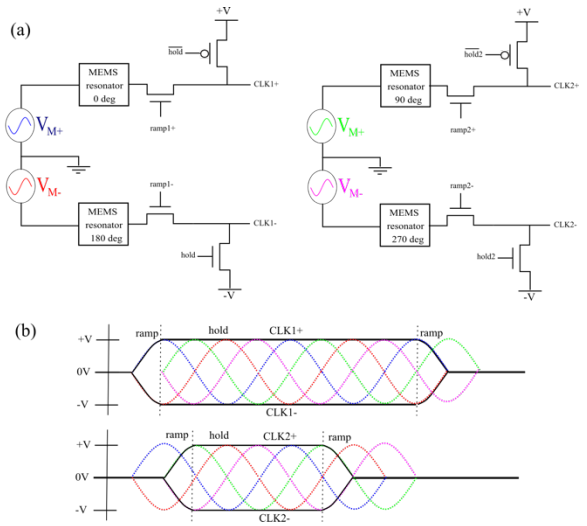

Fig. 9. (a) Circuit to generate multi-phased adiabatic clocks using MEMS resonators. (b) Timing diagram of clocking signals from MEMS resonators. [7]

The circuit shown in Fig. 9 utilizes four CMRs phased 90 apart. Enable transistors are implemented to select rising and falling edges while hold resistors are used to maintain valid logic levels.

B. Fabrication

The CMRs were fabricated on cleaved pieces of a four-inch AlN-on-Si wafer with a 1 µm AlN film. A 200 nm film of Al was sputtered and etched with BCl_3 and Cl_2. Multiple etch characterizations were completed during the fabrication process. The characterization of an AlN etch is shown in Fig. 10. Release was achieved using XeF_2. Five fabricated CMRs are shown in Fig. 11 following release. Fig. 12 shows a CMR that appears to have curled upward due to the stress in the AlN film. This curling has been documented in previous experiments [6], [7].

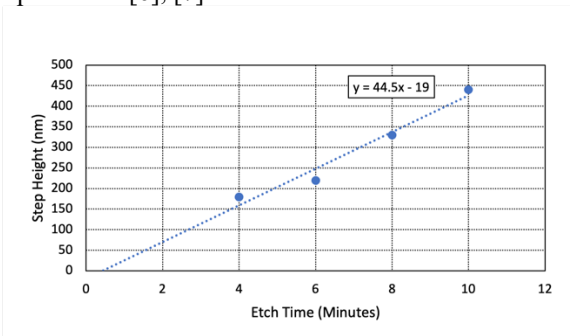

Fig. 10. Etch characterization of AlN using RIE recipe with BCL_3. An etch rate of 44 nm/min was determined.

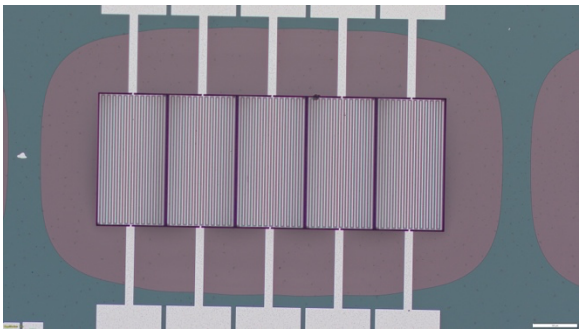

Fig. 11. Optical micrograph of five fabricated CMRs. The Si beneath the dark region surrounding the five CMRs was etched by the XeF_2 leaving only the AlN.

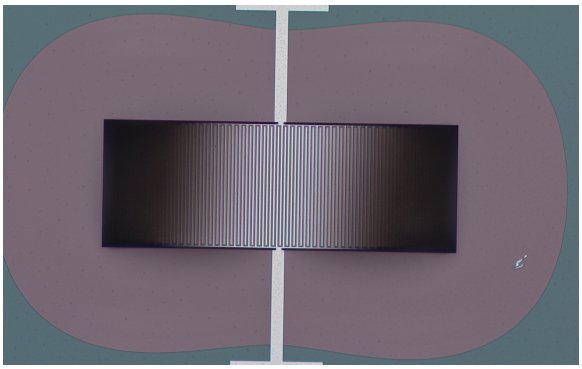

Fig. 12. Optical micrograph of CMR that appears to have curled upward at edges.

C. Conclusions and Next Steps

The fabricated CMRs require testing using radio frequency (RF) reflectometry. Following this testing, the CMR fabrication process will be refined for future fabrication. This will require etch characterizations for Al, AlN, SiO_2, and photoresist. Additionally, a new layout is desired to evaluate new CMR designs and to correct tolerance issues discovered in the current layout.

IV. Analysis of Coulomb Blockade in a Single-Electron Transistor Coupled to a Single-Electron Box

A. Background

Topological quantum computing is a promising area of research largely due to its intrinsic resistance to error from environmental perturbation. Our group is investigating single-electron transistors (SETs) for their potential to perform local charge sensing in topological quantum computing systems [8]. The following project aimed to analyze the Coulomb blockade diamonds generated by an SET when coupled to a single-electron box (SEB) acting as a trap.

An equivalent circuit model for the SET coupled to an SEB is shown in Fig. 13. The SEB acting as a trap produces periodic perturbations in the Coulomb blockade diamonds. The capacitance and conductance parameters of the SET and SEB were desired for analysis.

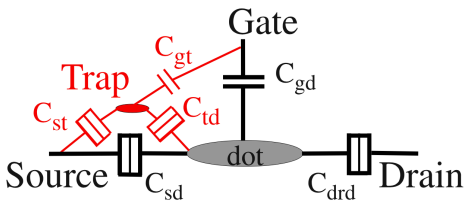

Fig. 13. Equivalent circuit for our model of a charge trap (red) coupled to a quantum dot (black). The trap occupation is 0 or 1 electron and it carries a very small current compared to the dot. The dot is treated within the orthodox model. [9].

B. Analysis

Analysis of the experimental Coulomb blockade diamonds shown in Fig. 14 was conducted through comparison with simulated results.

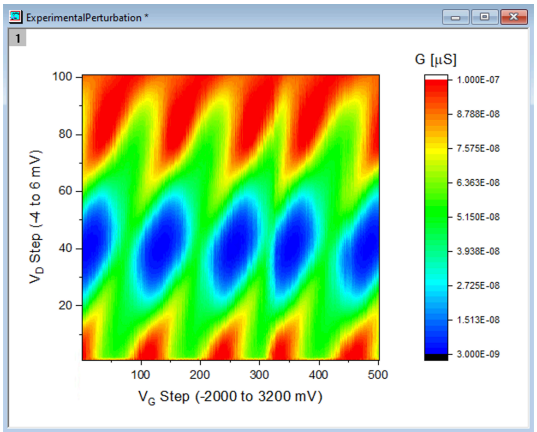

Fig. 14. Experimentally measured perturbed Coulomb blockade diamonds of an SET coupled to an SEB.

Initial efforts to replicate the experimental results through simulation focused on the unperturbed portion of the Coulomb blockade diamonds. The simulation of unperturbed diamonds was achieved by varying the capacitance and conductance parameters of an SET within a Matlab GUI [10]. This simulation is shown in Fig. 15.

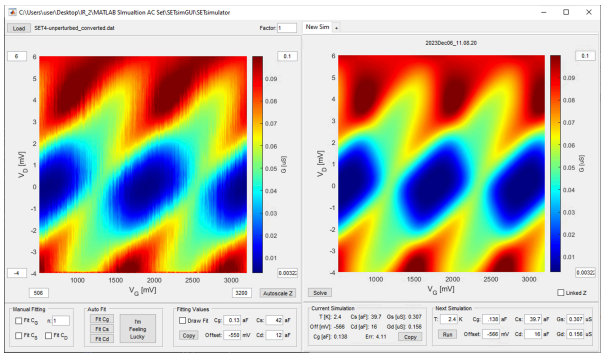

Fig. 15. Replication of experimental Coulomb blockade diamonds using Matlab GUI. Left) Experimental. Right) Simulated.

A Python script [10] was utilized to replicate the experimental perturbed Coulomb blockade diamonds through simulation. The perturbation was replicated by implementing the SET capacitance and conductance parameters from the unperturbed simulation. The capacitance and conductance parameters of the SEB were varied to position the perturbation properly. The results of this simulation are shown in Fig. 16.

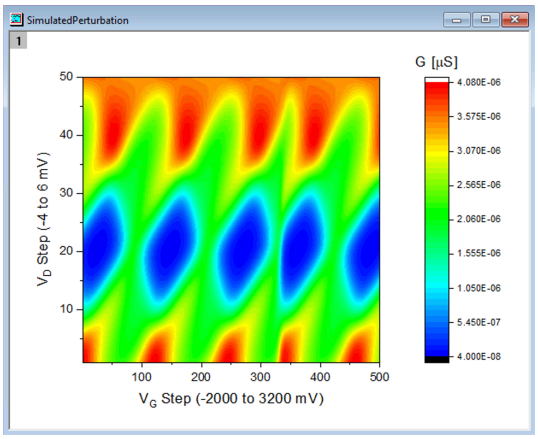

Fig. 16. Simulated perturbed Coulomb blockade diamonds of an SET coupled to an SEB.

This Python script was also utilized to demonstrate multiple periods of perturbation. This period is defined by the ratio in (1).

$$\frac{C_{gd}}{C_{gt}} \qquad (1)$$

In (1), C_{gd} is the gate to dot capacitance and C_{gt} is the gate to trap capacitance as shown in Fig. 13. The simulation shown in Fig. 17 utilizes a C_{gd} to C_{gt} ratio of 12 to generate perturbations every 12 diamonds.

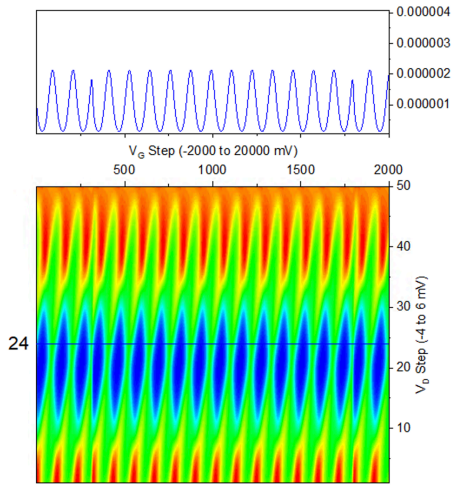

Fig. 17. Simulation of Coulomb blockade diamonds demonstrating multiple periods of perturbation. The period was defined to be 12 diamonds. Cgd=0.118 aF, Csd=39.7 aF, Cdrd=16 aF, Cgt=0.0098 aF, Cst=0.318 aF, Cdt = 0.98 aF, Gs=0.307 uS, Gd=0.156 uS.

C. Conclusions and Next Steps

The group intends to publish the simulations and corresponding parameters presented in this document soon. Additionally, following the success of these simulations, the group intends to analyze other experiments using similar techniques for parameter extraction.

V. References

[1] R. Celis-Cordova, A. O. Orlov, T. Lu, J. M. Kulick, G. L. Snider, "Design of a 16-Bit Adiabatic Microprocessor," *2019 IEEE International Conference on Rebooting Computing (ICRC)*, San Mateo, CA, USA, 2019, pp. 1-4, doi: 10.1109/ICRC.2019.8914699.

[2] Lent CS, Liu M, Lu Y, "Bennett clocking of quantum-dot cellular automata and the limits to binary logic scaling," *Nanotechnology*. 2006 Aug 28;17(16):4240-51. doi: 10.1088/0957-4484/17/16/040. Epub 2006 Aug 7. PMID: 21727566.

[3] Bannan, J, "Differential Driver and Evaluation Board and Programming Arbitrary Waveform Generators," Spring 2021

[4] McManus, R, "Arbitrary Waveform Generator Python Programming and Peltier Module Test Environment," Fall 2021

[5] McManus, R, "Programming of a Python GUI to Control Two Zurich HDAWGs and Layout of Three Printed Circuit Boards", Spring 2023

[6] McConnell, M. S. "Adiabatic Reversible Computing: Measurement of Heat Dissipation Using Nanothermocouples and Fabrication of MEMS Power Clocks." 2018 Dec; https://doi.org/10.7274/6682x34946r

[7] Cordova, R. C. "Reversible Computing: Adiabatic Capacitive Logic," 2022 Jul; https://doi.org/10.7274/qr46qz24b59

[8] Rahaman, MI, "Detection of Single Electron Charges in Nanoscale Dipoles and Anyon-Type Quantum Dots," 2023 Oct;

[9] H. C. George, M. Pierre, X. Jehl, A. O. Orlov, M. Sanquer, G. L. Snider, "Application of negative differential conductance in Al/AlOx single-electron transistors for background charge characterization," *Appl. Phys. Lett.* 25 January 2010; 96 (4): 042114.

[10] Pierre, M, "Simulation package for single electron devices," 2009 Jan 21;