|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | | | | | | |
|  | Draft | | |  | |
|  | Report | | | 1 (1) | |
|  | Date | Issue | Document ID | |
|  | 2025-10-27 | A |  | |
| Issued by | Classification Company Confidentiality | | | | |
| Adriaan Sadie | COMPANY RESTRICTED | | | | |
| Classification Export Control | Classification Other | | | | |
| NOT EXPORT CONTROLLED | NOT CLASSIFIED | | | | |

Multi-Voltage Thresholding Signal Acquisition with FPGAs

Table of Contents Page

[1 Background on MVT 3](#_Toc212556860)

[2 Objectives of Research 4](#_Toc212556861)

[3 Hardware Setup 4](#_Toc212556862)

[3.1 Full Final Hardware Setup 4](#_Toc212556863)

[3.2 Prototyping PCB Design 7](#_Toc212556864)

[4 Calibration 9](#_Toc212556865)

[4.1 DAC I2C Firmware Driver 9](#_Toc212556866)

[4.2 DAC Calibration Methodologies 11](#_Toc212556867)

[4.2.1 Triangle Wave Calibration 11](#_Toc212556868)

[4.2.2 Calibration by Metastability 12](#_Toc212556869)

[4.3 DAC Calibration Firmware 12](#_Toc212556870)

[4.4 DAC Setup and Initial Measurements 14](#_Toc212556871)

[4.4.1 Preparation 15](#_Toc212556872)

[4.4.2 Results 17](#_Toc212556873)

[4.5 DAC Compensation Setup 18](#_Toc212556874)

[4.6 Compensation Results 18](#_Toc212556875)

[5 ADC Acquisition Firmware 20](#_Toc212556876)

[6 Metastable Sample Region 21](#_Toc212556877)

[7 Results 22](#_Toc212556878)

[8 Conclusion 24](#_Toc212556879)

[9 Recommendations for Future Research 24](#_Toc212556880)

[10 Project Resources 24](#_Toc212556881)

[11 References 24](#_Toc212556882)

# Background on MVT

Multi-voltage thresholding samples an analogue signal in a manner analogous to flash-based analogue-to-digital converters (ADCs). The technique employs a voltage ladder to partition the input signal into discrete levels that can be encoded digitally. Conventionally, this is implemented on an integrated circuit using a resistor ladder. In this research, however, the internal Low-Voltage Differential Signalling (LVDS) ports of an FPGA are repurposed as comparators, with a simple external digital-to-analogue converter (DAC) generating the required reference voltages to achieve comparable functionality. The operating principle is illustrated in Figure 1.

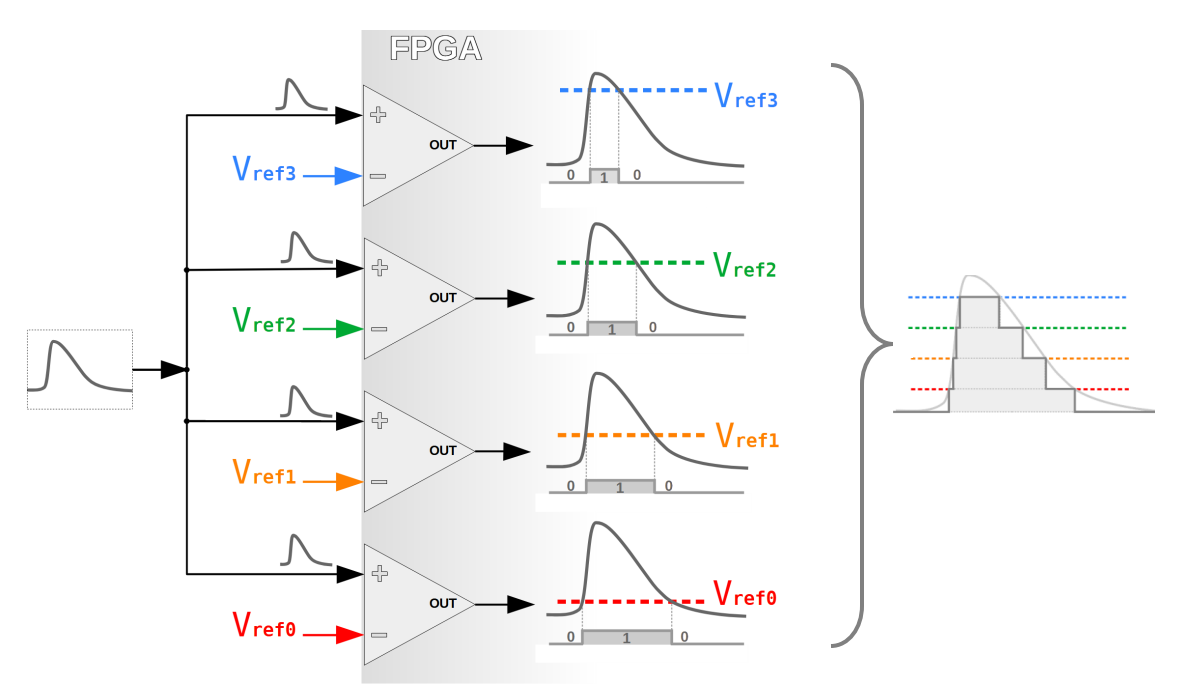


Figure 1: Concept of MVT using LVDS ports [1]

The primary advantage of this approach is its potential for extremely high sampling rates. With appropriate clock frequencies and I/O bank placement, rates up to 1 GSPS are attainable [1]. A significant drawback, however, is the exponential growth in resource requirements with resolution. An N-bit converter demands 2^N – 1 comparators, necessitating 2(2^N – 1) FPGA pins. For a 12-bit ADC, this equates to 4095 comparators and 8190 pins—clearly impractical for most devices. Nevertheless, the method may prove valuable in niche applications where speed is paramount and resolution requirements are modest.

# Objectives of Research

The primary goal of this research is to evaluate the feasibility of implementing the multi-voltage thresholding (MVT) technique on a Zynq UltraScale+ MPSoC device and to demonstrate its potential as a substitute for conventional ADCs in targeted applications. To achieve this, the following objectives have been defined:

1. Design and fabricate a printed circuit board (PCB) integrating the necessary DAC circuitry to interface seamlessly with the MPSoC evaluation kit.
2. Perform DAC calibration using the method described in [1] (elaborated in a later section).
3. Adapt an existing open-source IP core or develop a custom one to encode the LVDS comparator outputs into a 4-bit digital value.
4. Apply known analogue test signals to the fully integrated system and verify the IP core’s ability to accurately reconstruct the input signals in the digital domain.

# Hardware Setup

## Full Final Hardware Setup

At the start of the research, a decision was made to make use of a prototyping PCB rather than design and fabricate a custom one. The reason for this was that the principle was only to be proven, not to test the limits. Time and cost were also factors influencing this decision. For the purpose of this project, a hand-soldered prototyping PCB was sufficient. The final setup for testing can be seen in Figure 2. See Table 1 and Table 2 for brief descriptions of each component visible in the figure.

An XM105 Debug Card (shown in Figure 3) from Xilinx is used to expand the HP I/O pins of the MPSoC Evaluation Kit. The prototyping PCB with the DACs connects to this debug card via 40-way ribbon cable. The PCB also receives its power rails (3V3 and 1V8) from the XM105 via 6-way ribbon cable.

The HP 8112A pulse generator is used to calibrate the DACs before signal acquisition can take place.

The Fluke 179 multimeter is used for general measurements and to confirm DAC outputs.

The Tektronix MDO3024 oscilloscope is used for general measurements and debugging, as well as to generate a signal using its Arbitrary Function Generator (AFG) output.

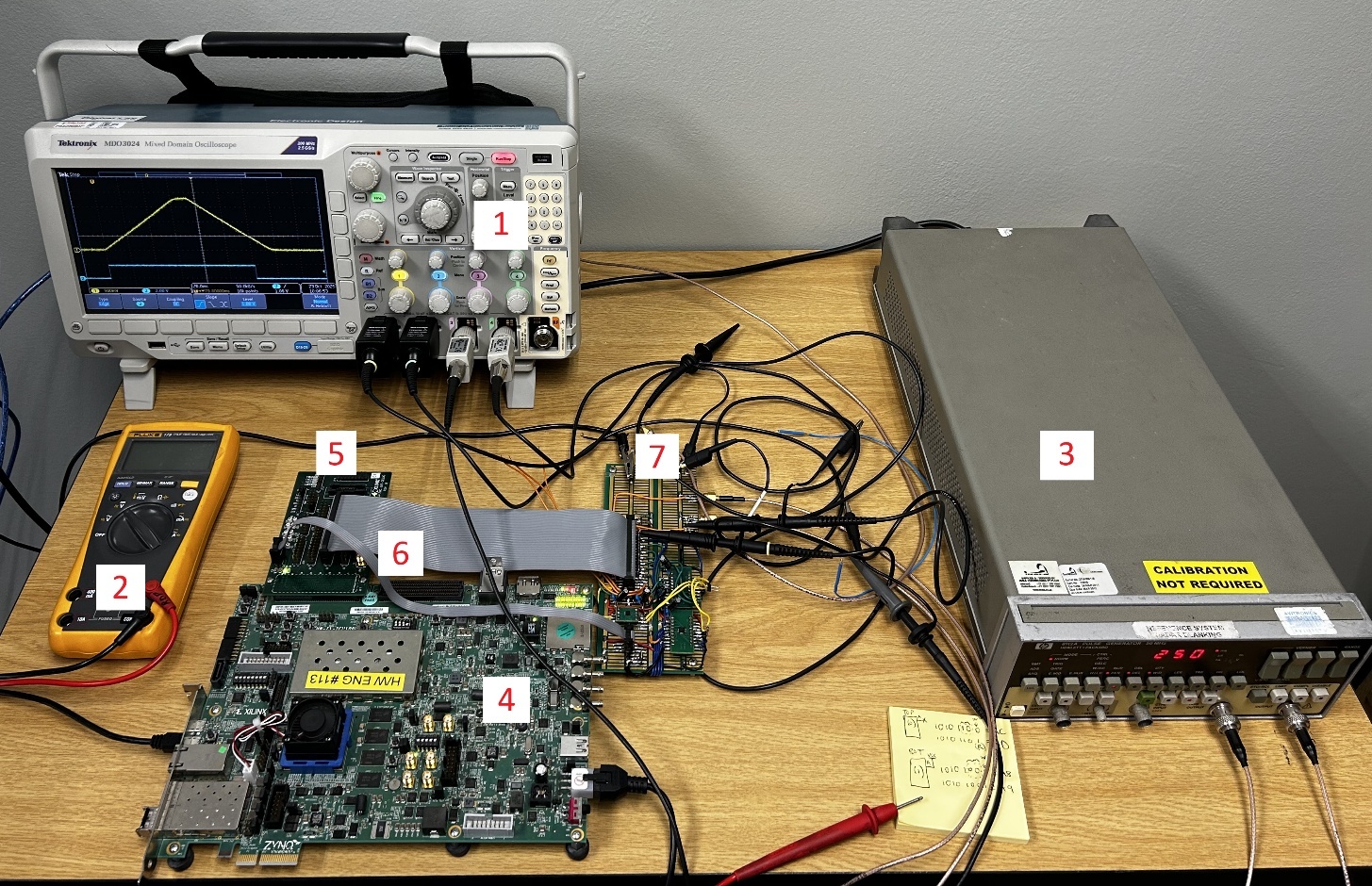


Figure 2: Full hardware setup

Table 1: Instruments used during project

|  |  |  |  |
| --- | --- | --- | --- |
| **Number** | **Instrument** | **Description** | **Function** |
| 1 | Tektronix MDO3024 | Mixed Domain Oscilloscope | Debugging, I2C protocol decoding, AFG output |
| 2 | Fluke 179 | Multimeter | Measuring the output from the DACs |
| 3 | HP 8112A | Pulse Generator | Generating the synchronisation pulse plus the Triangle waveform for DAC calibration |

Table 2: Hardware components used during project

|  |  |  |
| --- | --- | --- |
| **Number** | **Component** | **Description** |
| 4 | ZCU106 MPSOC Evaluation Kit | MPSoC Evaluation kit, containing the XCZU7EV-FFVC1156-2-E Zynq Ultrascale+ MPSoC device. Serves as the FPGA unit |
| 5 | FMC XM105 Debug  Card | Expander card that connects to FPGA Mezzanine Card (FMC) connector on the ZCU106 board. Provides access to a large number of High-Performance Input/Output (HP I/O) pins on the FPGA, particularly the LVDS pins. |
| 6 | 1x 40 Way Ribbon Cable; 1x 6 Way Ribbon Cable | Provides connectivity between the XM105 debug card and the Prototyping PCB |
| 7 | Prototyping PCB | Custom PCB that contains the external components, such as the 2x DAC devices as well as a voltage translator. |

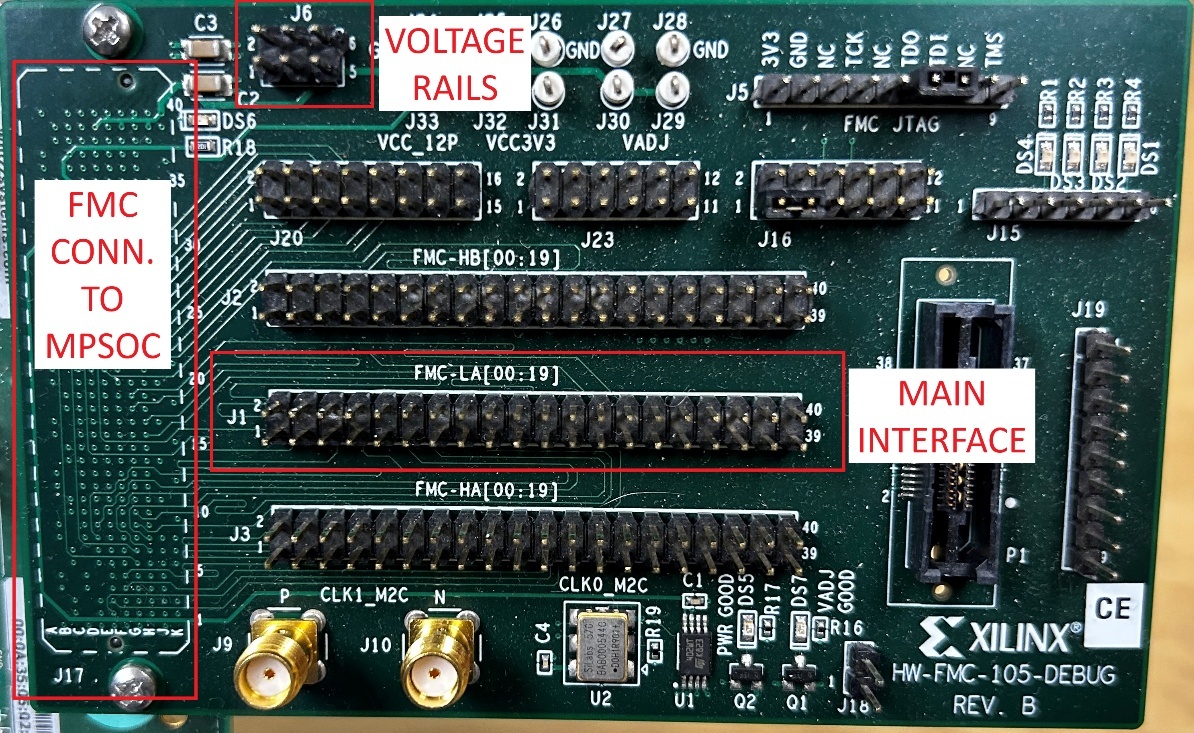


Figure 3: XM105 Debug board

## Prototyping PCB Design

The prototype PCB is depicted in Figure 4. To implement a 4-bit ADC using the MVT technique, 15 distinct voltage thresholds are required, necessitating 15 LVDS comparator inputs. The AD5669 octal DAC was selected for its eight independent voltage outputs; two such devices were employed to generate the full set of 15 reference levels.

The HP I/O banks of the Zynq UltraScale+ MPSoC operate at 1.8 V logic levels, whereas the AD5669 requires 3.3 V signalling. A TCA9802 bidirectional voltage-level translator was therefore incorporated to ensure compatibility.

Communication between the DACs and the MPSoC is established via I²C. The AD5669 provides a single address-configuration pin, which was used to assign the upper DAC (labelled 0 in the figure) an I²C address of 0xAC (binary 10101**10**) and the lower DAC (labelled 1) an address of 0xA8 (binary 10101**00**).

A 40-pin header interfaces with the XM105 expansion board’s J1 connector via a 40-way ribbon cable. A separate 6-pin header (labelled “Voltage Rails Header”) connects to the XM105’s J6 connector to supply the necessary power rails.

Three dedicated debug signals are routed from the MPSoC to the PCB to facilitate firmware development and troubleshooting.

Two SMA connectors are provided for external signal interfacing. The lower SMA accepts a synchronization pulse from an external pulse generator. The upper SMA serves dual purposes: during calibration, it receives a triangular waveform synchronized with the pulse input; in normal operation, it functions as the primary analogue input for the signal under measurement.

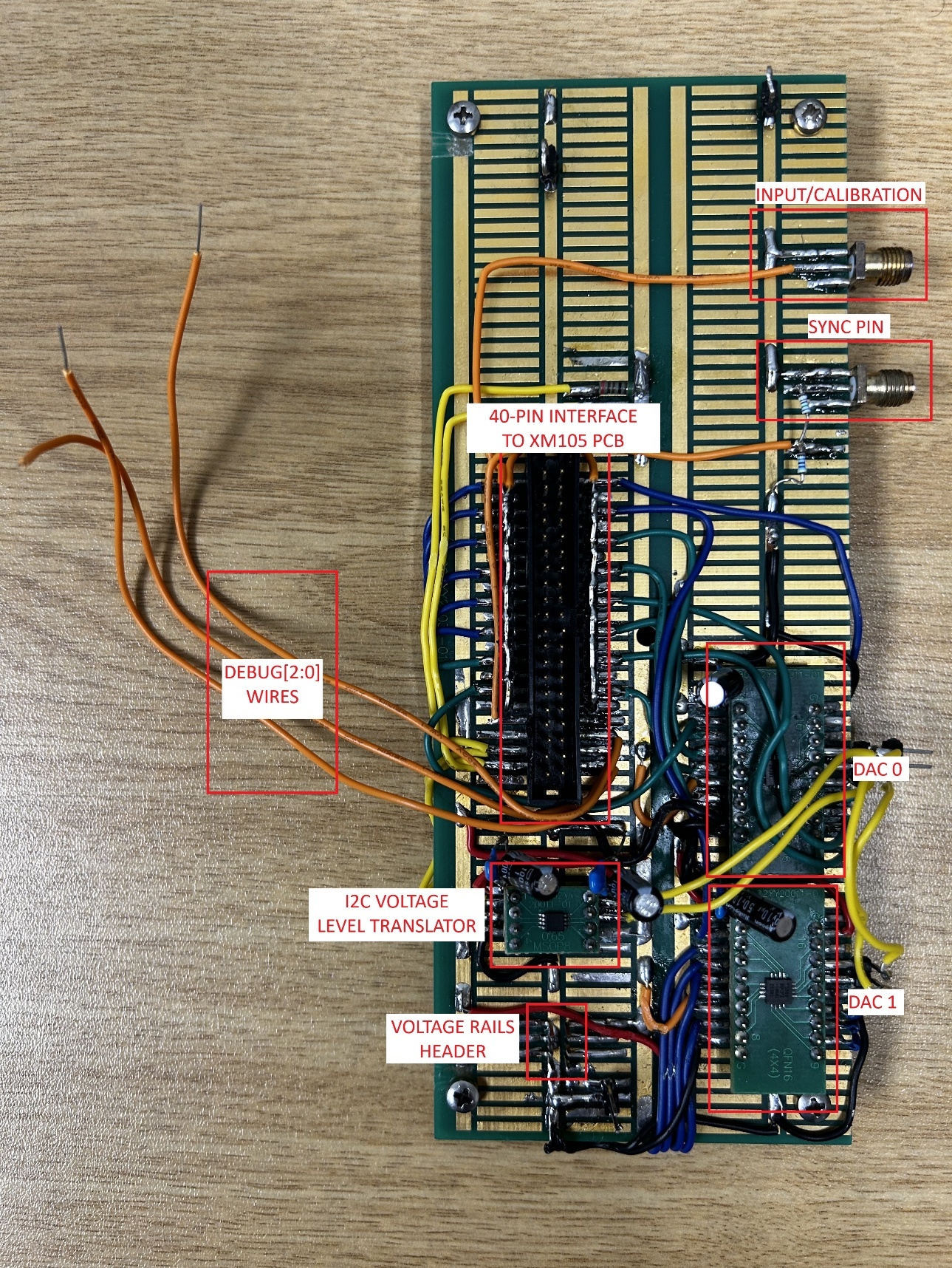


Figure 4: Layout of prototyping PCB

# Calibration

This section details the complete DAC preparation process required to enable accurate acquisition of analogue input signals for measurement. It begins with an overview of the I²C driver employed for DAC configuration, followed by the initial setup procedure. The calibration methodology is then presented, concluding with an analysis of the calibration results.

## DAC I2C Firmware Driver

A custom firmware driver was written to communicate between the DACs and the MPSoC. The firmware block (i2c\_master\_0) is shown in Figure 5.

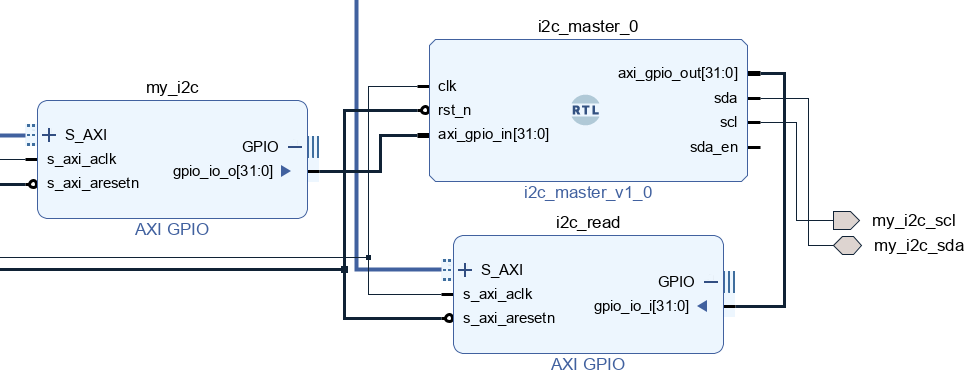


Figure 5: Firmware blocks for I2C interface

Using the Advanced eXtensible Interface (AXI) General Purpose I/O (GPIO) cores, data can be written (my\_i2c) to the i2c\_master\_0 core or read (i2c\_read) from it. By connecting the Joint Test Action Group (JTAG) to AXI IP Core as a secondary master to the AXI Interconnect, all AXI devices can be reached through the Vivado Hardware Manager Tool-Command-Language (TCL) Console.

The I2C master firmware is controlled through the AXI GPIO register (my\_i2c), which initiates an I2C transaction with the DACs. The control register bit-map is shown in Table 3. The firmware looks at the uppermost bit [31] to change from 0 to 1 (or rising edge). Then it clocks in the control register and initiates the transaction.

Table 3: I2C Master control register mapping

|  |  |
| --- | --- |
| **Bits** | **Description** |
| 31 | Trigger bit. When a 1 is read here after a previous 0, this transaction is sent through to DAC. I.e., first write all zeros to AXI GPIO, then write the actual value. |
| 30 | Write/Read. Write: 0, Read: 1 |
| 29:25 | Reserved |
| 24 | DAC ID. Top DAC (0): 0, Bottom DAC (1): 1. The addresses are hardcoded in the firmware |
| 23:16 | DAC Command |
| 15:8 | Upper data byte |
| 7:0 | Lower data byte |

The below example illustrates communication between the Vivado TCL console and the DACs. First a zero-vector command is created. Then a DAC enable command is created and finally a DAC value update command is created.

create\_hw\_axi\_txn wr\_zeros [get\_hw\_axis hw\_axi\_1] -address 80020000 -data 00000000 -type write

create\_hw\_axi\_txn wr\_enable\_1 [get\_hw\_axis hw\_axi\_1] -address 80020000 -data 804f0000 -type write

create\_hw\_axi\_txn wr\_dac1\_chE\_0V5 [get\_hw\_axis hw\_axi\_1] -address 80020000 -data 80344CE4 -type write

These commands are then executed one by one, with the zero-vector command in-between:

run\_hw\_axi wr\_zeros

run\_hw\_axi wr\_enable\_1

run\_hw\_axi wr\_zeros

run\_hw\_axi wr\_dac1\_chE\_0V5

The above commands are written to the DAC 0, and enables all 8 outputs (0x4F). Then it updates the value for channel E output to 0.5V (0x3 – change value command of DAC, 0x4 – Channel E, 0x4CE4 – corresponds to 0.5V given external reference of 1.8V).

## DAC Calibration Methodologies

From this point forward, the term DAC calibration will refer specifically to the compensation of inherent bias voltages in the LVDS input ports, which may deviate by up to 30 mV. The calibration procedure compensates for these offsets as follows: an initial reference voltage—generated by the DAC itself—is applied to the inverting input of each LVDS port, the resulting bias is measured using a designated calibration technique, and a corrective offset is then programmed into the corresponding DAC channel driving that inverting terminal. There are two suggested calibration methodologies, which are discussed below.

### Triangle Wave Calibration

The sync-pulse-plus-triangle method requires a signal generator capable of producing two synchronized outputs: a pulse train and a slow-ramping triangular waveform. The pulse train serves as a synchronization trigger, initiating a high-resolution timer within the firmware upon detection. The triangular waveform is applied to the non-inverting input of a selected LVDS comparator, while a known reference voltage - generated by the corresponding DAC channel - is supplied to the inverting input.

The firmware captures the timer values at both the rising and falling transitions of the LVDS output. These timestamps enable precise calculation of the comparator’s bias voltage. The operating principle is illustrated in Figure 6.

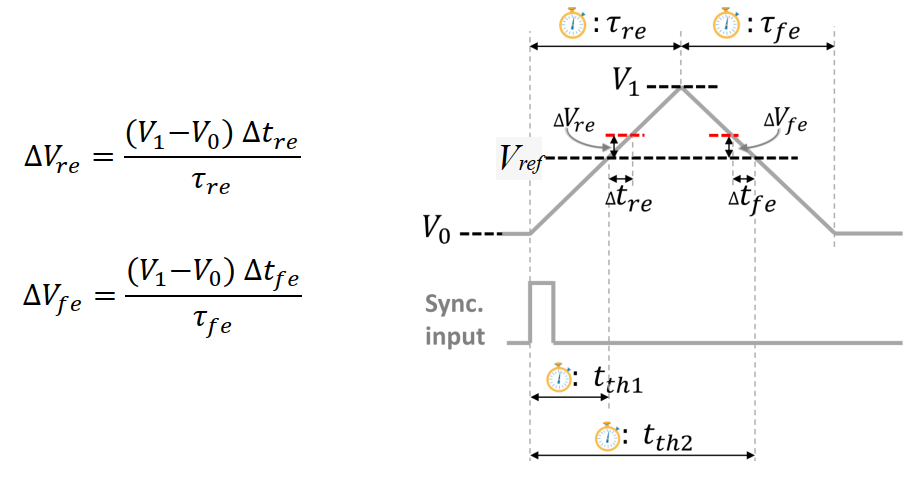


Figure 6: Triangle wave calibration illustration [1]

Due to time constraints, which restrict the research to a single calibration approach, the sync-pulse-plus-triangle method was selected for this project.

### Calibration by Metastability

The second approach, metastability-based calibration, eliminates the need for an external signal generator by exploiting the statistical behaviour of comparator metastability to quantify bias voltage. A known reference voltage from the DAC is applied to the inverting input of the LVDS comparator, while a quasi-static voltage—also DAC-generated—is supplied to the non-inverting input.

For each discrete step in the non-inverting voltage, the FPGA records the transition (switching) rate at the comparator output. The voltage step yielding the maximum switching rate corresponds to the point of highest metastability, enabling direct computation of the bias offset. This principle is illustrated in Figure 7.

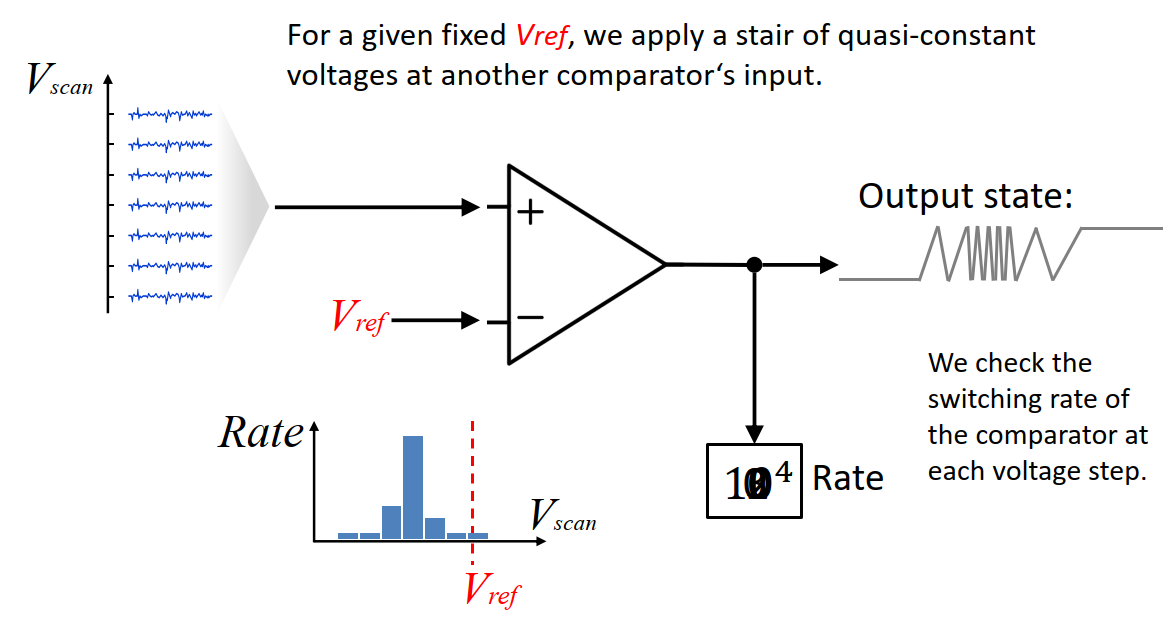


Figure 7: Metastability-based calibration [1]

## DAC Calibration Firmware

In order to perform the calibration procedure, the following firmware blocks were created:

* **adc\_front\_end**: Produces 16 single-ended signals by using the Differential Buffer primitive (IBUF\_DS) from Vivado on each of the incoming differential pairs. Also concatenates the 16 signals to form one single 16-bit logic vector. Refer to Figure 8.
* **dac\_cal**: Main calibration block. Receives control input from AXI GPIO register (dac\_cal\_control). Depending on the control input, the firmware block counts up to the voltage crossing threshold for a rising edge, or a falling edge. The bit (or LVDS comparator) to be measured can also be set with the control register. The first 2 bytes set the bit to be calibrated (e.g., 0x000B sets it to bit 11) and the last two bytes sets the counter to stop at the rising (0x000A) or falling (0x000F) threshold.
* **dac\_cal\_bram:** The dac\_cal block loads each count value one by one into this BRAM block. This data can then be read back using the AXI BRAM Controller (dac\_bram) and TCL commands.

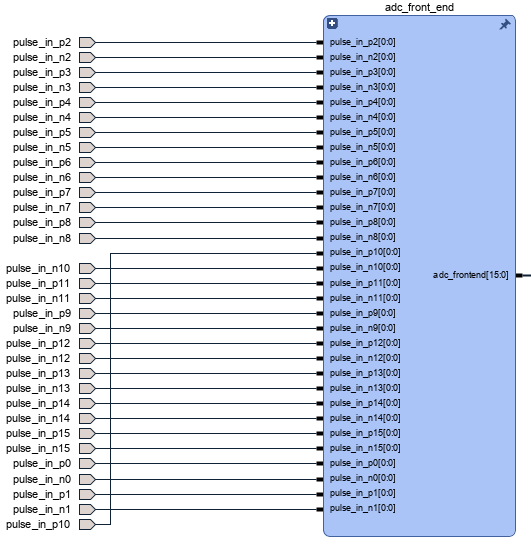


Figure 8: ADC Frontend firmware block

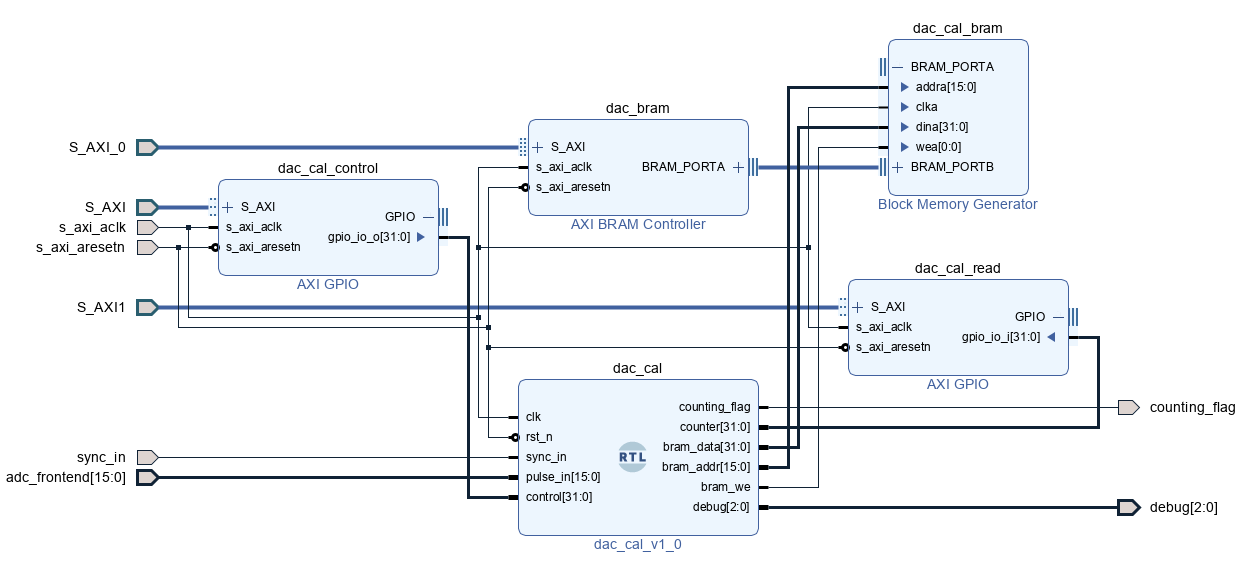


Figure 9: DAC Calibration firmware blocks

The following sequence illustrates the execution of a bias-voltage measurement for a single LVDS comparator at a specified reference level.

The target reference voltage is set to 0.9 V on Channel C of the second DAC. This value is programmed by writing the hexadecimal code 0x8000 to the corresponding DAC register. The measurement is configured to trigger on the rising-edge transition of the comparator output. A command is then issued to arm the firmware counter, which starts upon the next rising edge of the synchronization pulse.

The block RAM (BRAM) subsequently populates with timestamp data at a rate determined by the incoming pulse train. After an appropriate accumulation period, the BRAM contents are read and processed to compute the effective bias voltage. Since the BRAM communicates directly with JTAG to AXI interface, arming it with a zero-vector is not necessary before calling a read command.

create\_hw\_axi\_txn wr\_zeros [get\_hw\_axis hw\_axi\_1] -address 80020000 -data 00000000 -type write

create\_hw\_axi\_txn wr\_dac1\_chC\_0V9 [get\_hw\_axis hw\_axi\_1] -address 80020000 -data 81328000 -type write

create\_hw\_axi\_txn wr\_bram\_f [get\_hw\_axis hw\_axi\_1] -address 80030000 -data 0000000A -type write

create\_hw\_axi\_txn rd\_dac\_bram [get\_hw\_axis hw\_axi\_1] -address 84000000 -type read -len 128

Execute commands, then wait before reading the BRAM data:

run\_hw\_axi wr\_zeros

run\_hw\_axi wr\_dac1\_chC\_0V9

run\_hw\_axi wr\_bram\_f

after 32000

run\_hw\_axi rd\_dac\_bram

The output of the BRAM is 128, 32-bit values, or a 1024 hexadecimal string that looks like this:

0062fe840062de700062e3e8……

Each 8-character word is then split using a Python script, to produce 128 values in descending order. These values are copied to and Excel spreadsheet where the data is converted to decimal and used for further analysis.

## DAC Setup and Initial Measurements

The goal of this section is to determine the following:

1. The average bias voltage (or error) of each of the three bits
2. Determine the swing of error voltage of each of the three bits
3. Investigate whether the falling threshold lines up with the rising threshold (i.e., that there exists no hysteresis).

### Preparation

A suitable pulse generator (HP 8112A) was identified, capable of producing both a synchronization pulse and a quasi-triangular waveform. The latter is implemented as a standard pulse with an extended rise time, configured for a nominal leading-edge duration of 50 ms. Oscilloscope measurements indicate an actual rise time of approximately 60 ms, likely due to the pulse generator being long past its calibration interval. The sync pulse (blue) and quasi-triangular waveform (yellow) are shown in Figure 10. The sync pulse shows up as only an impulse because of the horizontal scale.

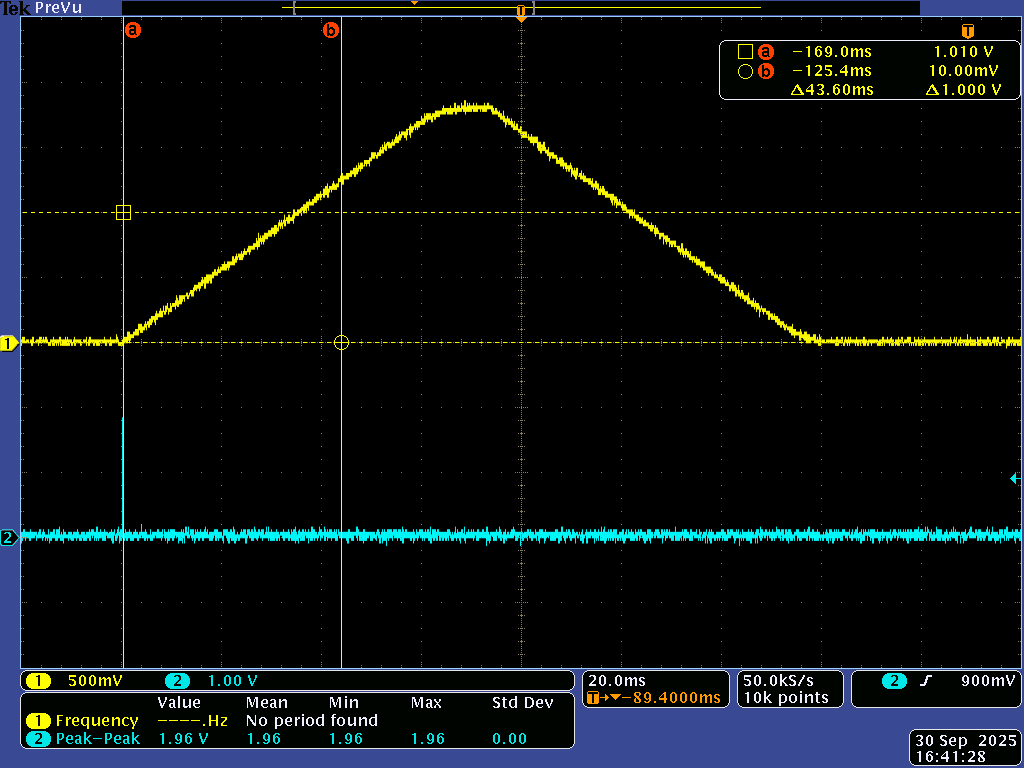


Figure 10: Triangle waveform and sync pulse

For the final signal acquisition, the calibration process needs to be applied to all channels across both DACs. However, only three LVDS comparators are shown in this report to demonstrate the calibration technique. The comparators will henceforth be called bits (corresponding to their place in the voltage ladder). Bit 0, bit 4 and bit 10 were chosen, so that there are 2 bits from DAC 0 (channels A and E) and 1 bit from DAC 1 (channel C).

To compute the bias voltage from the clock-count data retrieved from BRAM, the theoretical threshold-crossing time must first be established. Refer to Figure 10. The expected crossing time for a 1 V threshold is determined by placing cursor A at the start of the triangular waveform and adjusting cursor B to the point where the amplitude reaches 1 V. The resulting delta time, measured as 43.6 ms, represents the nominal duration from waveform initiation to the 1 V crossing.

When a 1 V reference is applied to the inverting input of the comparator, this 43.6 ms interval corresponds to the expected counter value at the moment of threshold crossing. Any deviation between the measured counter value and this theoretical count directly indicates the magnitude and polarity of the comparator’s bias voltage.

This method is used to calculate the threshold crossing times for three reference voltages: 0.4 V, 0.9 V and 1.4 V. The theoretical values of the clock counter (running at 200 MHz) are calculated in Table 4 and Table 5:

Table 4: Theoretical rising threshold crossing times

|  |  |  |  |
| --- | --- | --- | --- |
| DAC Voltage | Measured time from sync pulse to threshold crossing (ms) | Slope | Theoretical Threshold Count Value @ 5 ns CLK |
| 0.4 | 14.00 | 27.98507 | 2800000.00 |
| 0.9 | 32.40 | 27.98507 | 6480000.00 |
| 1.4 | 49.40 | 27.98507 | 9880000.00 |

Table 5: Theoretical falling threshold crossing times

|  |  |  |  |
| --- | --- | --- | --- |
| DAC Voltage | Measured time from sync pulse to threshold crossing (ms) | Slope | Theoretical Threshold Count Value @ 5 ns CLK |
| 0.4 | 122.60 | 27.91667 | 24520000.00 |
| 0.9 | 104.20 | 27.91667 | 20840000.00 |
| 1.4 | 87.40 | 27.91667 | 17480000.00 |

The slope of the triangular waveform was calculated separately for the rising and falling segments by dividing the measured voltage delta by the corresponding time delta. These values should theoretically be equal in magnitude and opposite in sign.

### Results

The results for the three chosen bits are show in Table 6 and Table 7.

Table 6: Rising threshold voltage errors

|  |  |  |  |
| --- | --- | --- | --- |
| LVDS Port | Voltage Reference | Average Error (mV) | Average Max Swing (mV) |
| Bit 0 | 0.4 V | 10.96 | 4.54 |
| 0.9 V | 28.6 | 4.58 |
| 1.4 V | 12.82 | 5.91 |
| Bit 4 | 0.4 V | -14.29 | 3.80 |
| 0.9 V | 17.28 | 3.45 |
| 1.4 V | 2.46 | 4.47 |
| Bit 10 | 0.4 V | 15.6 | 4.25 |
| 0.9 V | 20.73 | 3.92 |
| 1.4 V | 9.82 | 5.11 |

Table 7: Falling threshold voltage errors

|  |  |  |  |
| --- | --- | --- | --- |
| LVDS Port | Voltage Reference | Average Error (mV) | Average Max Swing (mV) |
| Bit 0 | 0.4 V | 42.26 | 4.67 |
| 0.9 V | 26.89 | 4.93 |
| 1.4 V | 48.14 | 4.42 |
| Bit 4 | 0.4 V | 60.01 | 5.09 |
| 0.9 V | 34.14 | 3.43 |
| 1.4 V | 55.44 | 3.85 |
| Bit 10 | 0.4 V | 70.55 | 4.26 |
| 0.9 V | 41.11 | 3.18 |
| 1.4 V | 59.66 | 4.33 |

The results show that there is a very small swing between the minimum and maximum voltage errors (roughly 4 mV). The data also seems to show that the error for the falling threshold is much larger than for the rising threshold. According to [1], this should not be the case. The unexpected results in Table 7 may be because of measurement errors.

Another thing to note about the results, is that different voltage levels produce different bias voltages for each comparator (or bit).

## DAC Compensation Setup

To demonstrate the compensation, only the 0.9 V reference voltage is used and only rising thresholds are counted. Each average error (or bias voltage) is added to 0.9 V, as shown in Table 8. The corresponding hexadecimal value is then written to the specific channel of the specific DAC and the measurement process is repeated.

Table 8: Compensation values for DAC

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Compensation** | | | | |
| Bit | Error (0.9V) | New Value: | New DAC decimal Value | Hex Value |
| 0 | 28.60 | 0.9286 | 33809 | 8411 |
| 4 | 17.28 | 0.9173 | 33397 | 8275 |
| 10 | 20.73 | 0.9207 | 33523 | 82F3 |

## Compensation Results

The compensated results for bit 0, bit 4 and bit 10 are shown in Figure 11, Figure 12 and Figure 13, respectively. For both the pre- and post-compensated measurements, the values are split into 20 bins and plotted on a graph.

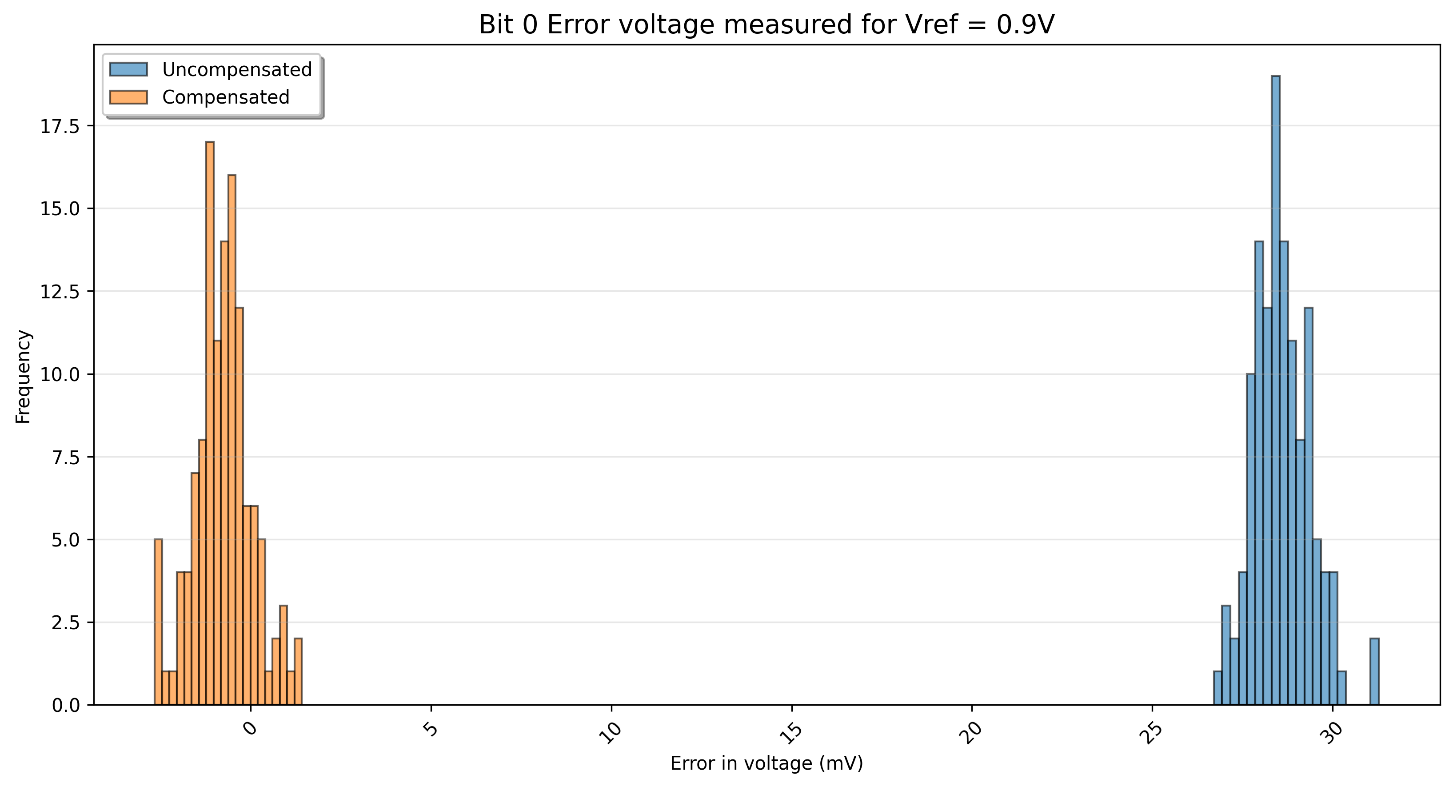


Figure 11: Bit 0 error voltage compensation

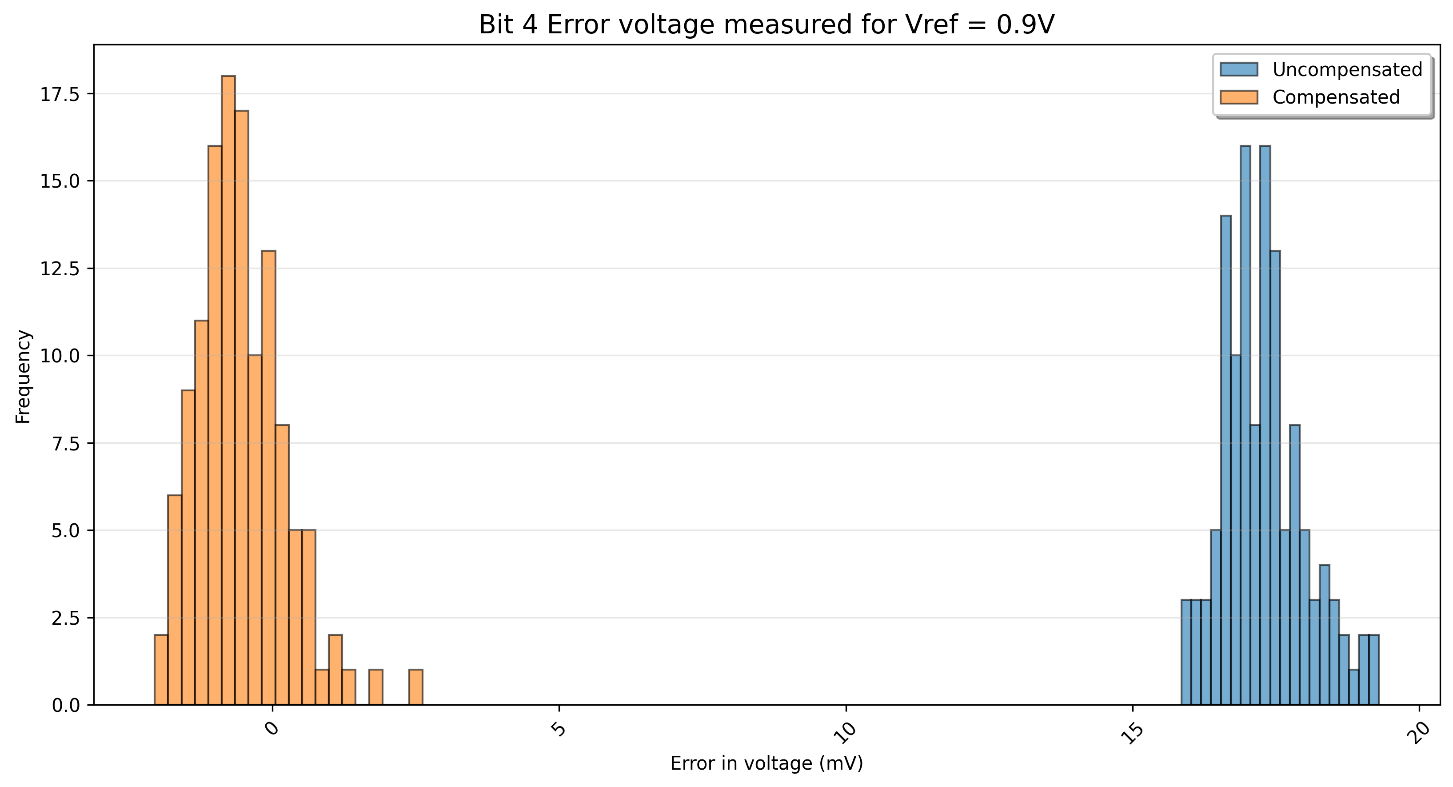


Figure 12: Bit 4 error voltage compensation

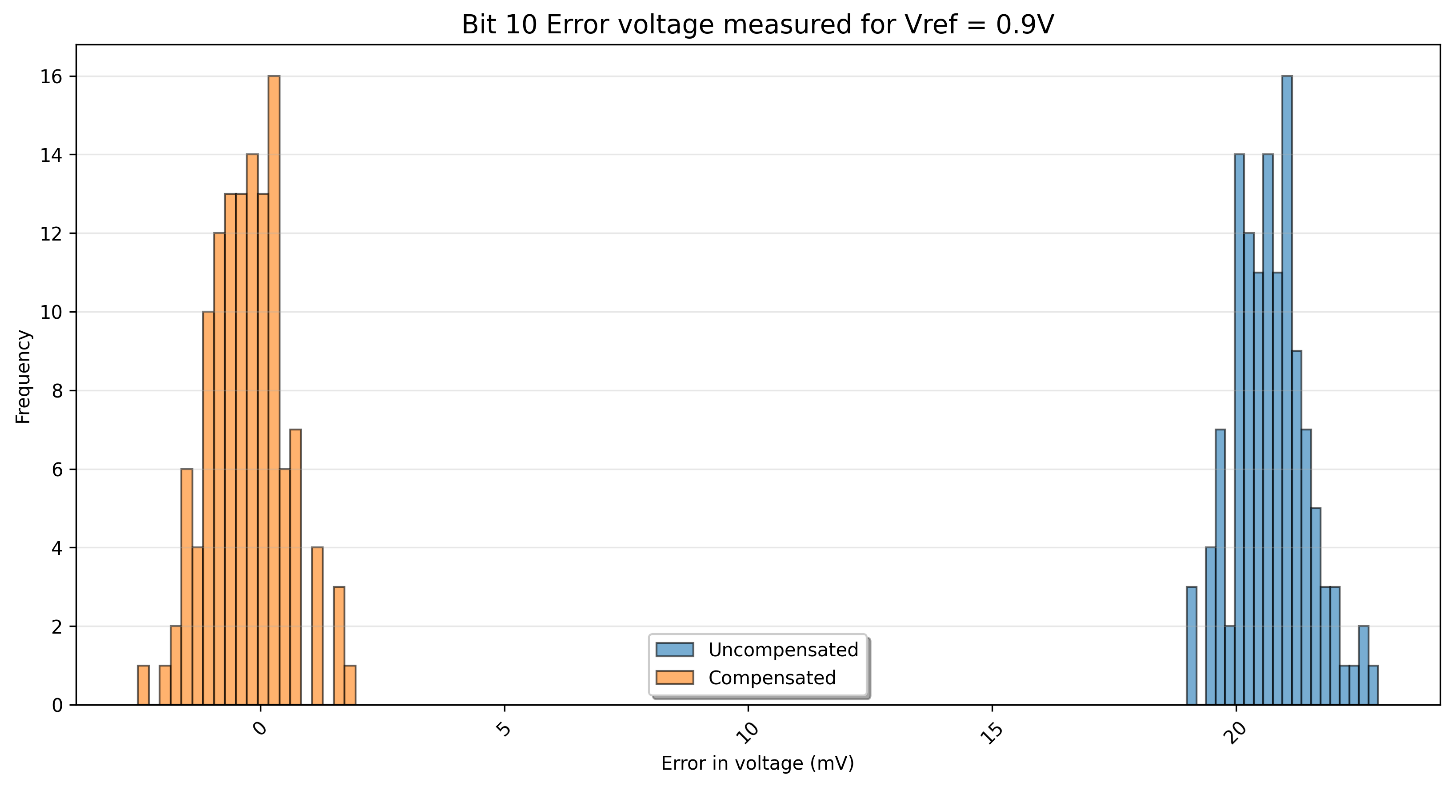


Figure 13: Bit 10 error voltage compensation

The figures show a clear improvement and an almost complete removal of bias voltage after compensation.

# ADC Acquisition Firmware

In [1], an open-source “MVT-quad” firmware core is given that is capable of measuring a 4-comparator input signal (2-bit ADC). The core uses a fast clock to timestamp rising and falling edges on each comparator, then reconstructs the analogue signal with the timestamp information.

Initially the goal of this research was to expand on the open-source core and turn it into a 4-bit ADC. However, it was decided to develop a new firmware block that more closely resembles a Flash-ADC. The firmware blocks are fairly simple and shown in Figure 14. The blocks are briefly described below:

* **adc\_encoder**: Simply encodes the 16 bits from the comparators to form a 4-bit ADC value.
* **adc\_to\_bram\_0**: On each clock cycle until the BRAM block is full, write the encoded value of the ADC to BRAM.
* **adc\_bram\_control**: Initiates data capture (rising edge on bit 31)

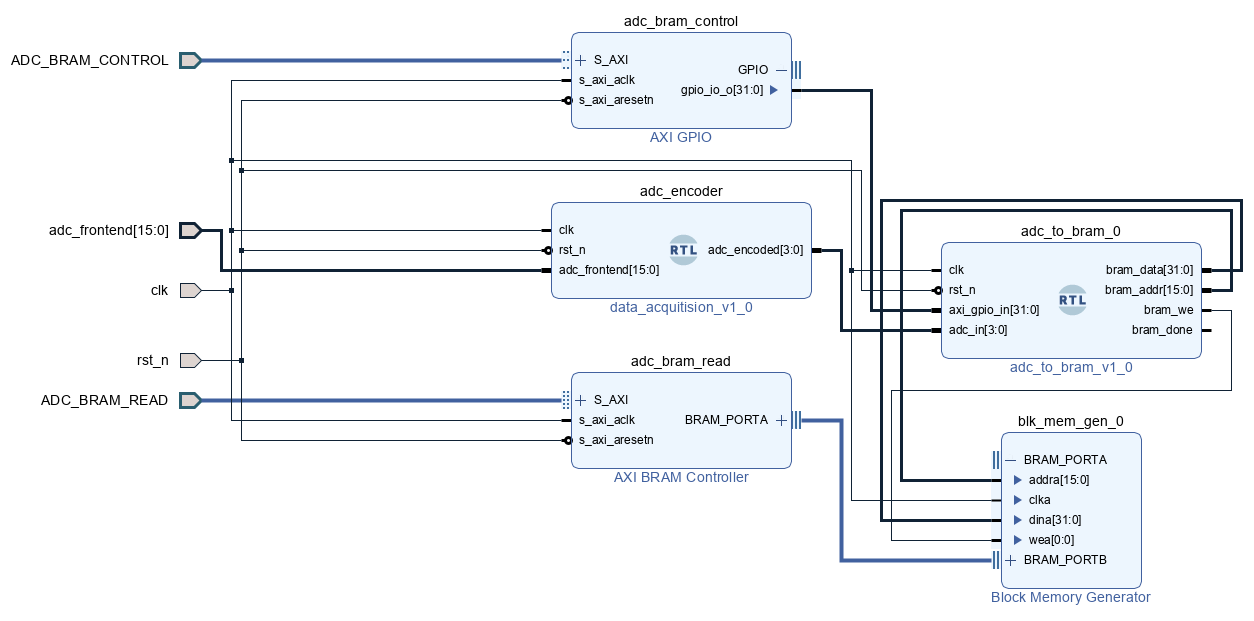


Figure 14: Data acquisition firmware blocks

# Metastable Sample Region

There exists a small window around each comparators reference voltage where the resulting output will be metastable. The sampling instant can coincide with this metastable region, resulting in incorrect data. However, this is only true for one comparator at a time. This principle is shown in Figure 15.

Logically then, the amount of these errors will increase for an increase in sampling frequency.

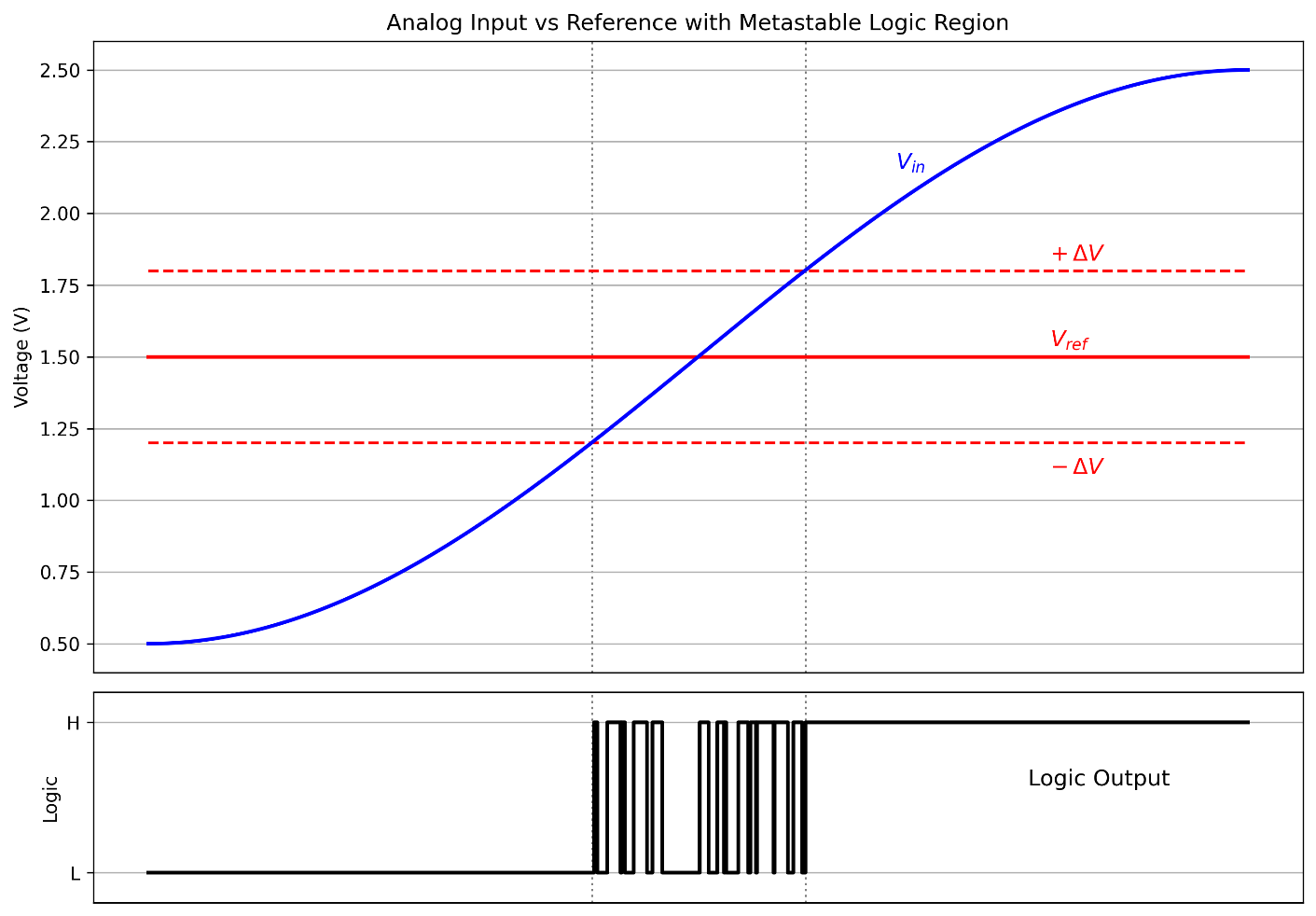


Figure 15: Metastable sampling error

# Results

A 500 kHz sinusoid with amplitude 850 mV and offset of 850 mV was generated using the Tektronix MDO3024 oscilloscope’s AFG output to fall within the 1.8V range of the MVT ADC.

The resulting ADC data was captured in BRAM, and plotted alongside the AFG sinusoid input in Figure 16. The sampling frequency is the same as the system clock of the firmware (200 MHz).

Notice the “glitch” at the beginning of the plot, around 0.25 V. This is due to the metastable sampling error.

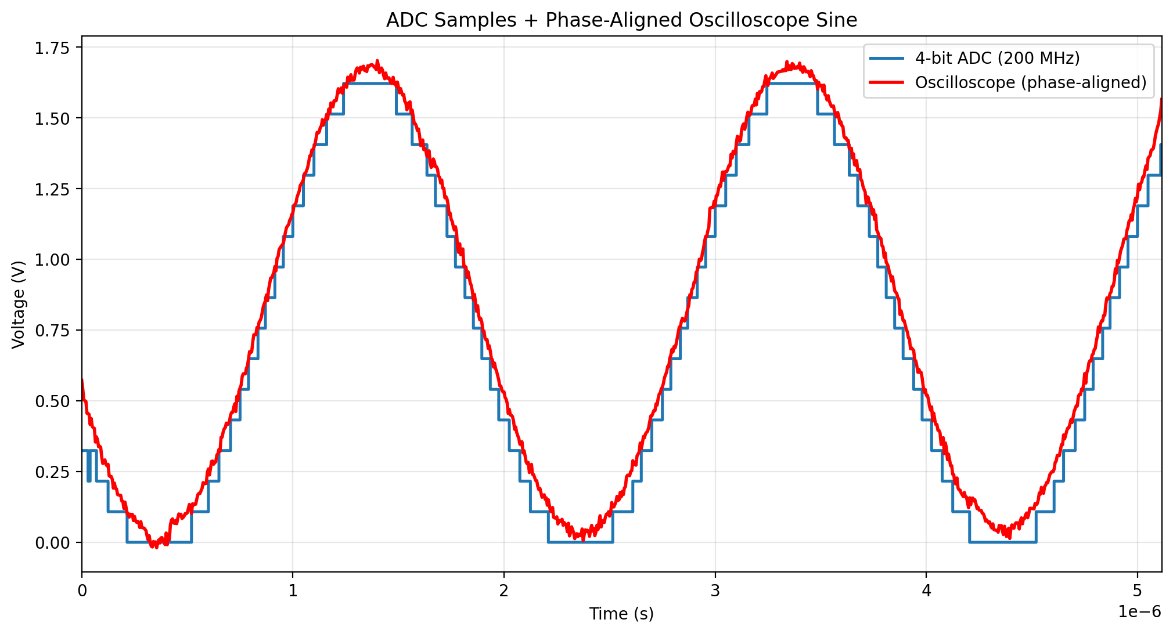


Figure 16: 4-Bit ADC output at 200 MHz sampling frequency

By increasing the system clock to 400 MHz, the sampling rate can be increased as well, as shown in Figure 17. The input sinusoid frequency was also increased to 1.75 MHz.

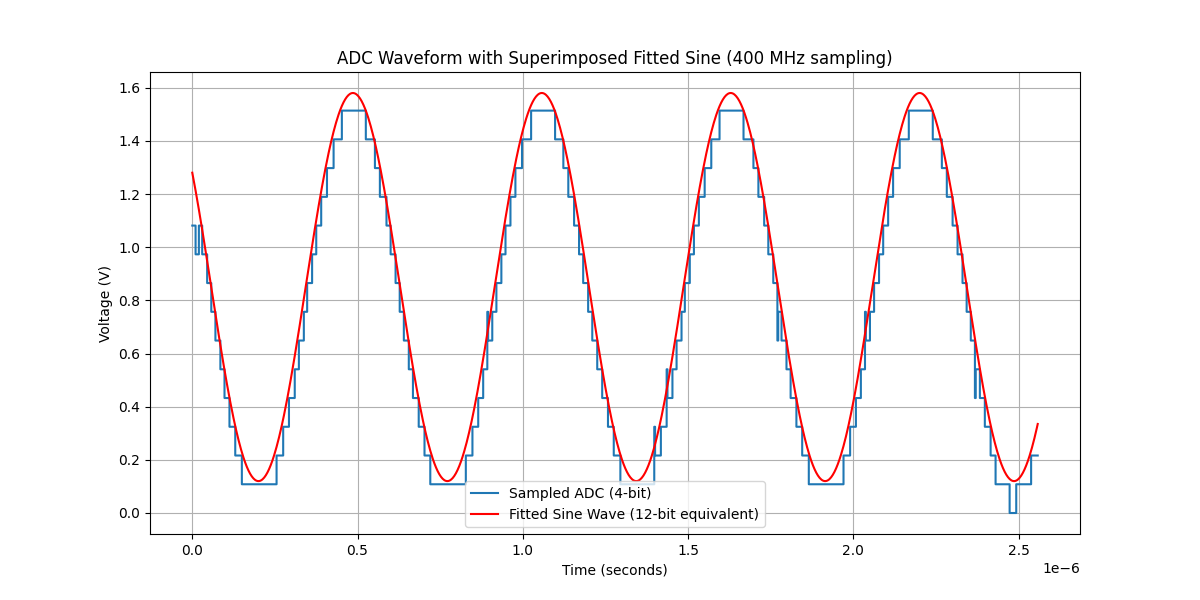


Figure 17: 4-Bit ADC output at 400 MHz sampling frequency

As expected, the number of “glitches” increased with an increase in sampling frequency.

# Conclusion

# Recommendations for Future Research

# Project Resources

# References

[1] Eliseev, D., Ehlert, E., Hebbeker, T., Merschmeyer, M., & Presser, C. (2024). An open-sorce IP-Core for Multi-Voltage Thresholding signal acquisition with FPGAs. *Journal of Instrumentation*, *19*(02), C02035.