Application note

Interleaving 2 pieces of ADQ35 to get 20 GSPS and stream data to GPU

Table of Content

1 Introduciton 3
2 Applications 4
3 System descritpion 4
3.1 Data streaming 5
3.2 Hardware details 5
3.3 Signals to the two ADQ35 digitizers 5
3.4 Software Details 6
3.5 Example code 6
3.5.1 calibrate.py 7
3.5.2 interleave.py 7
3.6 Additional measurement equipment 8
4 results 8
4.1 Time domain 8
4.2 Frequency domain 9

1 INTRODUCITON

Demonstrated in this application note[1] is:

- Time interleaving of two 10GSPS 12-bit digitizers (ADQ35) to achieve 20GSPS.
- A timing error better than 4ps over the input frequency range of DC – 2.5GHz.
- Each digitizer streams the data at 12.8GB/s over PCIE to a GPU.

A block diagram of the demonstration set-up is in Figure 1Figure 2. A photo of the 2 pieces of ADQ35 in in Figure 2. Example code is available upon request from SPD_Support@Teledyne.com.

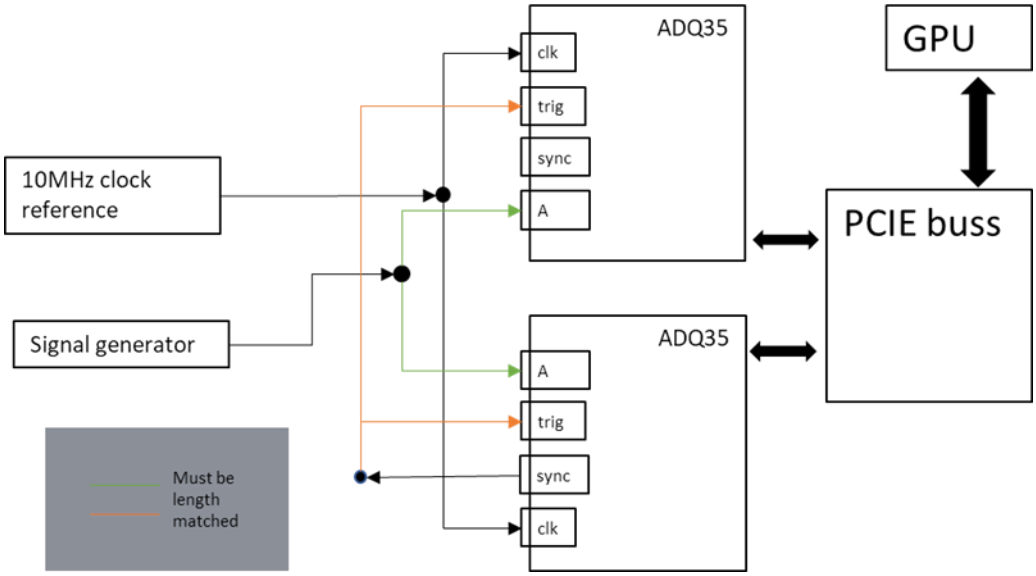

Figure 1 Block diagram.

[1] Note that this application note describes one test and the result of that test. The application note is not a binding specification of the products. Please refer to the datasheet for product specification. Availability and performance of third-party component is not guaranteed.

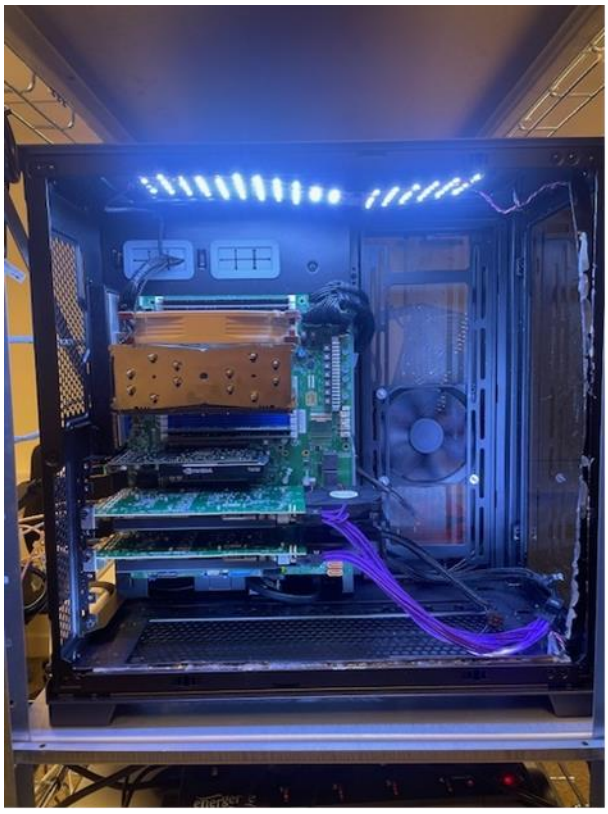

Figure 2 Photo of 2 pieces of ADQ35 inside the PC

2 APPLICATIONS

The ADQ35 is a high-speed digitizer suitable for sampling fast pulses and high frequency signals. The ADQ35 operate up to 10 GSPS. However, sometimes a faster sampling is required. We demonstrate a system where 2 ADQ35 units are time interleaved to reach up to 20 GSPS.

This system is useful in analytical instruments, RF receivers, time-of-flight applications and many more high-speed systems.

We also demonstrate how to handle the enormous amount of data that is produced. This is a non-trivial task for which we present methods.

3 SYSTEM DESCRITPION

We let the two digitizers sample the split analog input signal in a time interleaved way. This is achieved by, for one of the digitizers, use the onboard capability of delaying/skew the incoming reference clock.
Post processing of the acquired data will be performed on a GPU. While a vector optimized CPU implementation of the signal processing might suffice computationally, we motivate the choice of a GPU for the tasks by their comparatively significantly higher memory bandwidth.

3.1 Data streaming

Key considerations in choosing a system are PCIE and GPU memory bandwidth. The PCIE bandwidth can be broken down into how many PCIE lanes the CPU has and how the lanes are routed on the motherboard.
One ADQ35 can stream data at up to 14 GB/s through its 16 lanes gen 3 PCIE interface.
A more recent GPU can be found with 16 lanes gen 4 PCIE interface. The jump in generation to gen 4 provides a 2x bandwidth increase.
Given this, the system must have one 16 lanes gen 4 and two 16 lanes gen 3 PCIE ports.

3.2 Hardware details

The system used contains:

- AMD EPYC 7282 CPU, providing 128 gen 4 PCIE lanes.
- Supermicro H12SSL-I motherboard, providing five 16 lanes gen 4 PCIE ports.
- Nvidia A2000 GPU, providing 288 GB/s VRAM bandwidth and gen 4 x16 PCIE interface.

3.3 Signals to the two ADQ35 digitizers

To be able to control the phase difference of the sampling clocks between the digitizers, a split 10Mhz clock reference is connected to the clk-connector of each digitizer. The phase of the clocks is tuned by a built-in fine tune delay line in each ADQ35.

To be able to distribute a single trigger command to both units, we let the software command SWTrig() create a pulse out on the sync-connector. The output from the sync-connector of one digitizer is split and connected to the trig-connector of each digitizer. The trigger port of each digitizer is then configured to synchronize the trigger event to its 10Mhz clock reference. This provides a way to start acquiring data simultaneously on both cards.

The cable connections are shown in Figure 3.

The connectors and how to set them up is described in ADQ3 Series Digitizers – Users Guide.

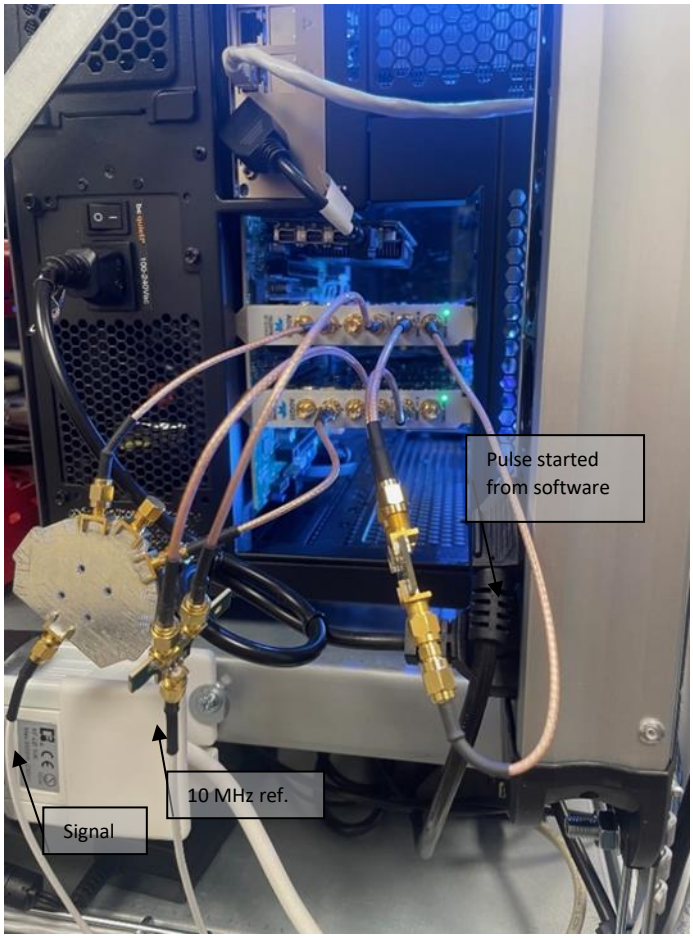

Figure 3 Digitizer Connections

3.4 Software Details

- Operating system is Ubuntu 20.04.6 LTS.
- Release Package 2024.1 of digitizer firmware and ADQAPI
- Nvidia driver 550.54.14, CUDA 12.4

3.5 Example code

The software part, calibrate.py and interleave.py, are based of the two python example scripts provided with the TSPD release package. They are found in the folder examples/python/data_readout and examples/python/data_transfer_gpu_nvidia.
The data_readout script is our most basic example. It uses the simplified data_readout functionality of the API to retrieve data from the digitizer. See the file README in the example for further details.
Script data_transfer_gpu_nvidia uses a lower-level API functionality to move data from the digitizer to a Nvidia GPU. It also has its own README with details.

Key changes required in both scripts are:

- Extend to support two digitizers.
- Configure digitizers to use 10Mhz reference clock.
- Set clock_system.delay_adjustment_enabled.
- Use trigger port to create events to start acquiring records. Synchronize these events to 10Mhz reference clock.
- (For one card) Configure a Pulse Generator to start on software trigger and output the pulse on the sync-connector.
- (For one card) Configure clock_system.delay_adjustment.

3.5.1 calibrate.py

One way to find a value for clock_system.delay_adjustment, which makes the two cards sample the signal 180 degrees out of phase to each other, is to provide a known sine signal. Fit a sine function to the sampled data from both digitizers. From these functions compute the required delay. This is implemented in calibrate.py which is based on the standard example data_readout.py. The flow chart of the operation is in Figure 4.

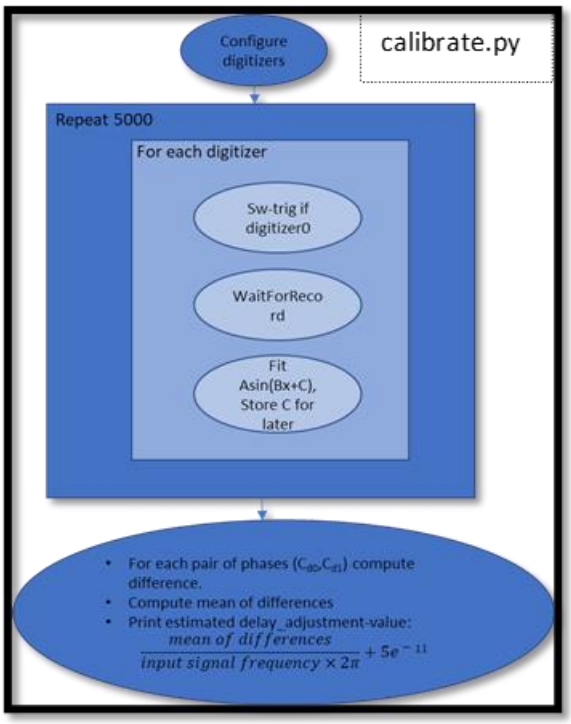

Figure 4 Flow chart calibrate.py

3.5.2 interleave.py

The script uses Data Transfer to move data from the two digitizers directly into the memory of the GPU. Different from calibration.py is that the SWTrig() call does not immediately start the Pulse Generator. Instead, it starts a Periodic Event Source which starts the Pulse Generator that create the

output on the sync-connector. This, since we require both cards being fully configured and ready when the trigger signal becomes active.

3.6 Additional measurement equipment

Additional equipment used:

- Stanford Research Systems CG635, as 10Mhz clock reference.
- Agilent Technologies N5181A, as Signal source

4 RESULTS

4.1 Time domain

Figure 5Figure 5 depicts a sampled signal before and after delay_adjustment-calibration.
Figure 6 depicts the measured time error. I.e., deviation from the expected half a sampling period, 50ps, time difference between the cards measured at different input frequencies.

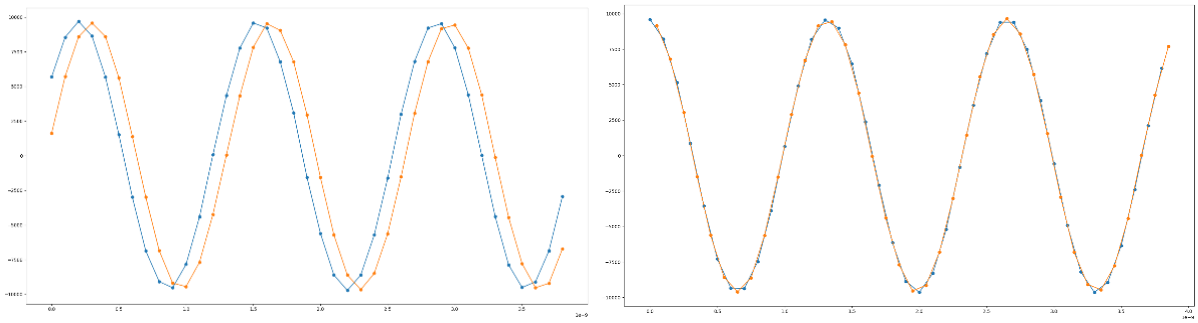

Figure 5 Orange are sample points from digitizer0, blue from digitizer1. Y-axis is signal amplitude in ADC-codes, X-axis is time in s.Left plot is before delay_adjustment calibration. Right, after calibration.

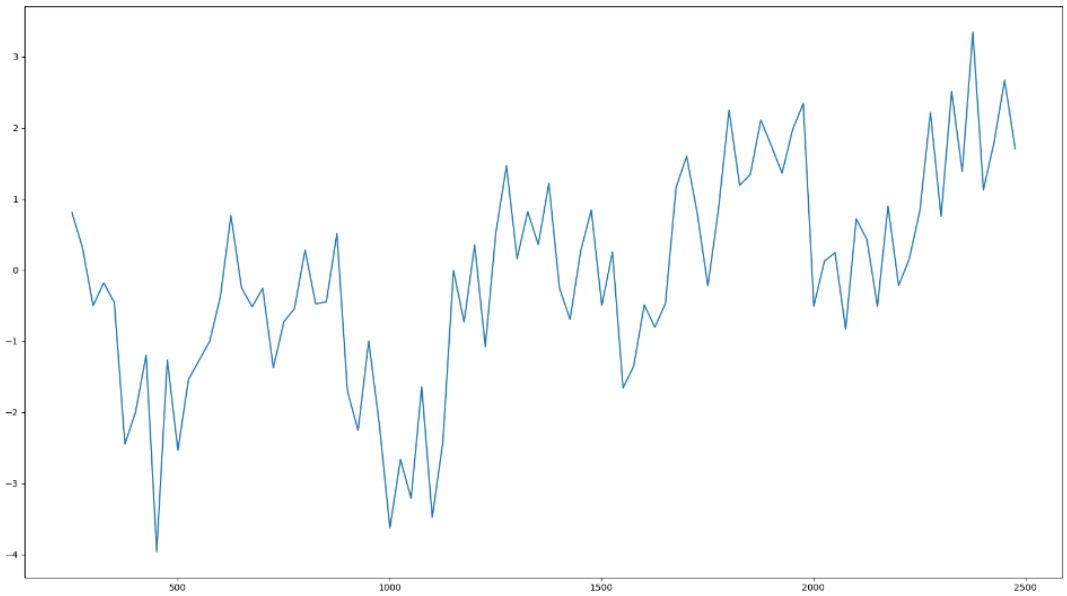

Figure 6 X axis is test-tone frequency in MHz, Y axis is the measured time error in ps.

- 20GSPS is reached with a time error of less than 4ps between the two channels.
- Streaming to a GPU and there interleaving data in cupy at a rate of up to 25.6GB/s was measured.

4.2 Frequency domain

The FFT of the test tone at 2 GHz is shown in Figure 7. The FFT contains the interleaving spurs from the interleaving of two ADQ35. Interleaving spurs are cause by time (phase), amplitude and offset mismatch. Figure 8 contains the same signal compensated for offset and amplitude errors. This is then the contribution from the time error to the interleaving spur.

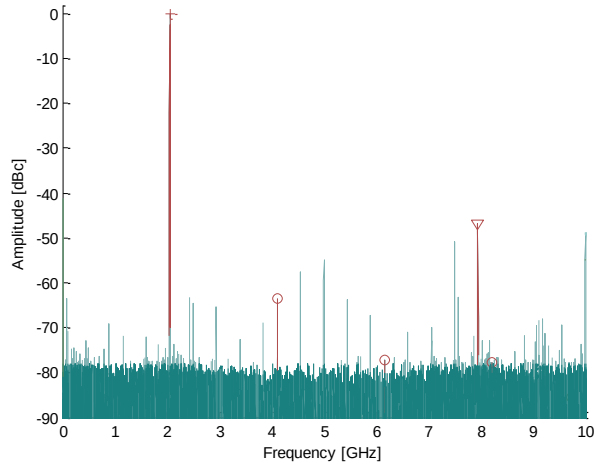

Figure 7 FFT of 2 GHz signal sampled at 10 GSPS

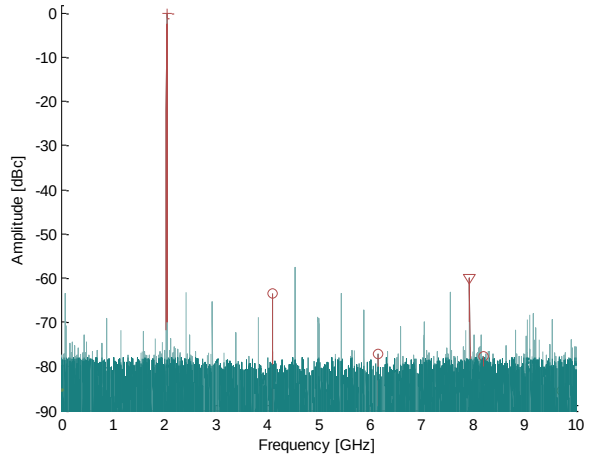

Figure 8 FFT, compensation for Amplitude error.