ADQ14 Hard X-Ray Monitor

Wojciech Walewski

DMCS

Contents

1 Introduction **3**

1.1 Document structure . . . 3

1.2 ADQ14 . . . 3

2 SPDevices DevKit **4**

2.1 The concept of User Logic . . . 4

2.2 Samples in the FPGA . . . 5

2.3 Data flow in User Logic 1 . . . 5

2.4 Data flow in User Logic 2 . . . 5

2.5 Data paths in the FPGA . . . 6

2.5.1 DMA . . . 6

2.5.2 User Registers . . . 6

2.6 Bypassing User Logic . . . 7

3 Custom Firmware **8**

3.1 Pulse Detection and Timing Module . . . 9

3.2 Pulse Shaping Module . . . 11

3.3 Pulse Sampler . . . 11

3.4 Spectrum Storage and Transfer . . . 12

4 Using Custom Firmware **14**

4.1 QADQScope . . . 14

4.2 Spectrum Dialog . . . 15

A User Logic Data Bus **16**

A.1 User Logic 1 . . . 16

A.2 User Logic 2 . . . 16

A.3 User Logic 2 Registers . . . 17

1 INTRODUCTION

1.1 DOCUMENT STRUCTURE

This document describes the functionality of a firmware package developed for the ADQ14 digitizer. The solution implements a real time Pulse Height Analyzer for exponential pulses, intended for use in Hard X-Ray Monitors. The Pulse Height Analyzer firmware is built on top of the SPDevices ADQ14 DevKit. Initial chapters of this paper go over the digitizer board itself, highlighting its features and limitations. The custom firmware solution is built on top of a development kit provided by SPDevices with ADQ14.

1.2 ADQ14

For information on the digitizer board itself refer to the manufacturers' website. Specifically the datasheet and the user manual.

2 SPDEVICES DEVKIT

The custom spectroscopy firmware is built on top of a development kit provided with the digitizer board. The base system is referred to as the DevKit by the manufacturer. A full documentation of the DevKit can be downloaded from the manufacturer's website. An offline copy is included with the firmware source code in the `documentation` folder.

2.1 THE CONCEPT OF USER LOGIC

The modules provided with the DevKit control the Analog Digital Converter, triggering, packeting, the DRAM data queue and the PCIe interface. The subsystems responsible for these parts of the pipeline are provided in the form of encrypted IP cores and cannot be modified. Two modifiable modules, referred to as User Logic 1 and 2, are exposed in between, enabling the use of entirely custom algorithms at two points in the data flow. User Logic 1 is placed right before the trigger control, enabling the use of detection or smoothing filters with the digitizer's level trigger feature. User Logic 2 allows for modifications being placed right before the individual consecutive samples are packed into records, assigned metadata and queued in the DRAM for transmission to the host computer. The DevKit structure, with the placement of the User Logic modules highlighted in gold, is shown in Figure 1. Both modules are described in greater detail in the next chapters.

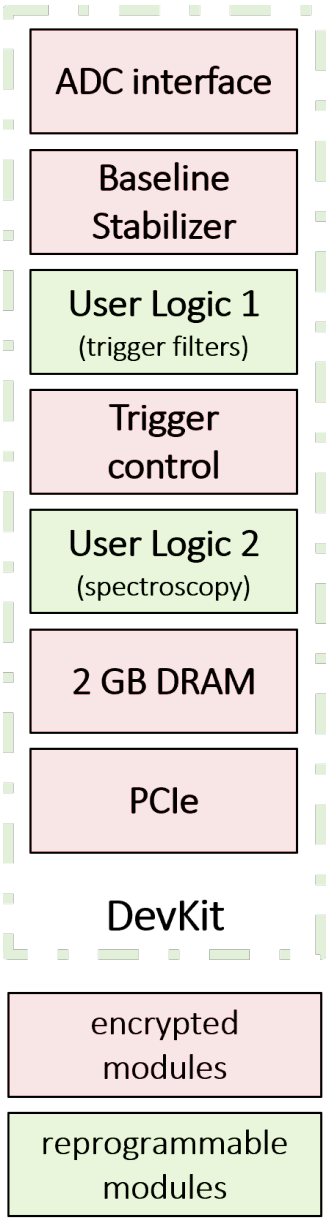

FIGURE 1: DEVKIT OVERVIEW

2.2 SAMPLES IN THE FPGA

The digitizer's ADC is capable of sampling the signal at a rate up to 1 GS/s or 1 GHz. The FPGA is clocked at 250 MHz, resulting in a parallel design where each channel produces 4 new consecutive samples for the FPGA on each clock cycle. The ADCs produce 14-bit samples. The raw signal is however subject to a digitial gain and shift. Two additional fractional bits are appended to the 14-bit samples before calculations. Two configurations are factored in the gain and shift calculations. The board comes preconfigured with a factory calibration that is always applied. A second configuration is made available to the users and can be freely modified from base values to apply custom digital signal shift or change the gain.

2.3 DATA FLOW IN USER LOGIC 1

After the digital gain and shift, the 16-bit samples are packed into a data bus and made available at the User Logic 1 module. This component is located before the level trigger. By default the raw sample data is passed on to the triggering module unchanged. If the device trigger mode is set to the level trigger, digital filters can be implemented in User Logic 1 to modify the trigger behaviour. In initial testing a configurable moving average smoothing filter was placed in this module to reduce the influence of noise. At one point a boxcar and a trapezoidal filters were successfully implemented for pulse detection. These filtering modules were created in an abstract manner and can be combined to modify the level trigger behaviour to a desired shape and signal-to-noise ratio.

The samples that are passed to the level trigger module can differ from those that are sent further down the processing pipeline, meaning that it is possible to trigger on appropriately filtered data, while still storing unmodified samples at the host PC. The current version of the firmware does not perform any filtering in this part of the firmware. All spectroscopy logic has been moved to User Logic 2, however the filter modules remain usable and can be reintroduced if the need appears.

2.4 DATA FLOW IN USER LOGIC 2

User Logic 2 deals with data that has passed through User Logic 1, decimation and shifting to accomodate for pretrigger (horizontal shift). User Logic 2 is responsible for tagging the start and end of a single acquisition window (record). Further DevKit modules can then then pack and queue the records for transmission. User Logic 2 also inserts header data containing the metadata. The record length is fixed by default and set in the control application on the host PC.

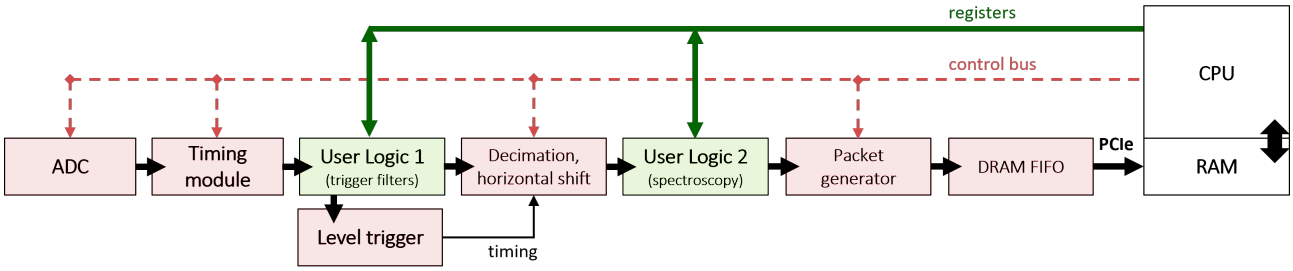

FIGURE 2: PRIMARY DATA FLOW IN THE FPGA

2.5 DATA PATHS IN THE FPGA

Figure 2 shows the primary data flow in the FPGA. Records built of the 16-bit samples, together with the metadata are initially stored in the digitizer's internal DRAM. This memory forms a FIFO queue, that is periodically transferred over to the host PC. The ADQ14's RAM has a capacity of 2 GB. With two bytes per sample up to a billion samples can be queued for data transfer before overflow leads to data loss. With up to a billion samples being generated every second for every active channel, this memory must be efficiently transferred. Two data paths exist within the firmware for exchanging data with the host PC.

2.5.1 DMA

After being packed in records and queued in the digitizer's DRAM, the acquired samples are sent using Direct Memory Access. This path makes it possible to write data directly to the host PC's RAM, without the need of CPU involvment. Naturally, the host PC's RAM is not unlimited, so the data must be processed (e.g. saved to a hard drive) at a rate comparable to the DMA transfers. DMA is a one-way path, so data can be transferred from the digitizer to the host PC, but not the other way around. Any data can be inserted in place of the samples for each channel to leverage the fast data path. The custom Pulse Height Analyzer relies on this fact to periodically transfer computed spectras.

2.5.2 USER REGISTERS

A substantially slower two-way data path is available through the user registers (top data bus on Figure 2). Both User Logic modules implement a small individually addressable memory block, that can be written to or read from by both the host PC and the FPGA. The memory structure contains 32-bit integers. 2^{14} addressable words are available in User Logic 1, and 2^{19} in User Logic 2. First 4 positions in both modules are reserved for internal use by the DevKit and cannot be modified in the custom firmware. The custom PHA firmware relies on User Registers primarily for transferring configuration settings to the FPGA. The current version of the firmware also enables spectra transfer through this data path for testing purposes.

2.6 BYPASSING USER LOGIC

Both User Logic modules can be bypassed using the application available on the host PC. When User Logic 1 is bypasssed, the incoming raw signal is fed directly to the board trigger. Any filters placed there to improve the triggering performance will not work. When User Logic 2 is bypassed no signal data is transferred. Any logic placed there, like the spectroscope, will not output valid data.

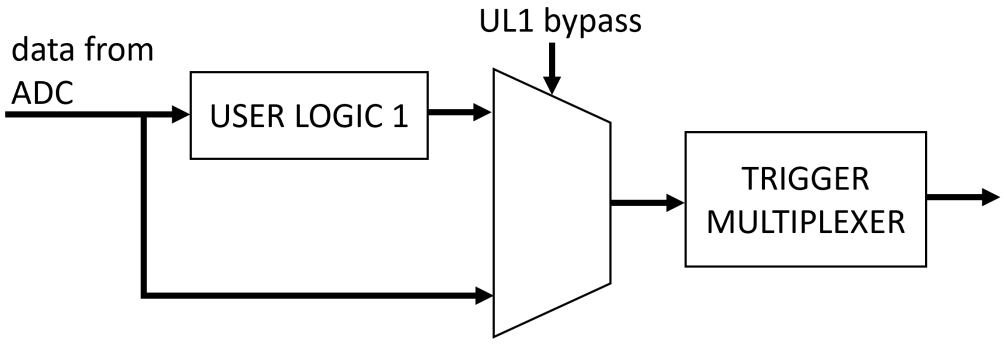

FIGURE 3: BYPASS IN USER LOGIC 1

3 CUSTOM FIRMWARE

The hardware spectroscopy solution consists of four primary subsystems:

- Pulse Detection and Timing Module
- Pulse Shaping Module
- Pulse Sampler
- Spectrum Storage and Transfer

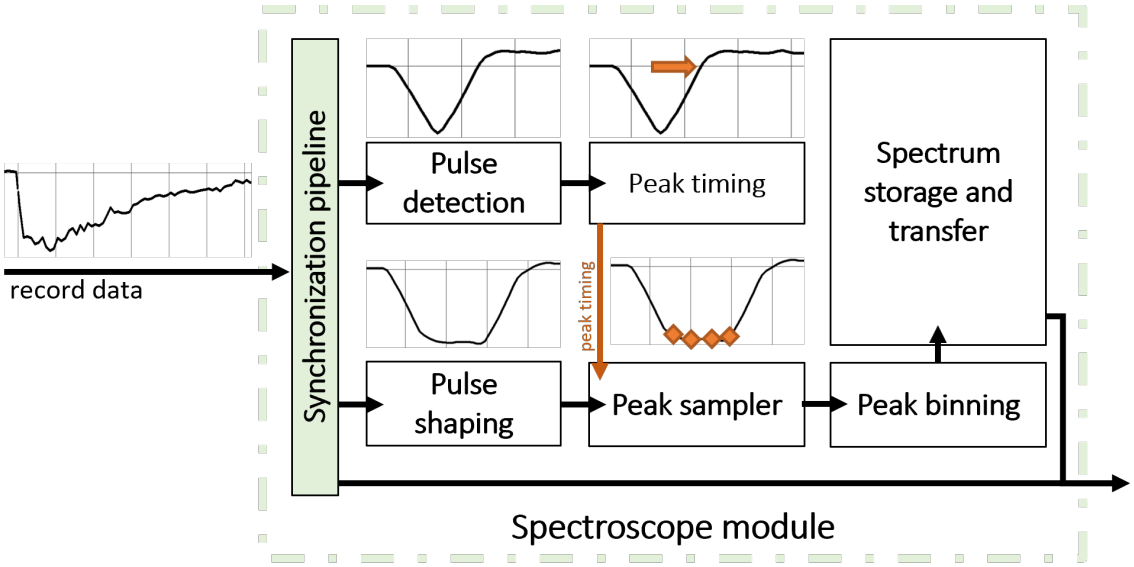

FIGURE 4: SPECTROSCOPE MODULE STRUCTURE

The entire logic is located within the User Logic 2 module. Figure 4 presents the internal structure of the spectroscopy module and how the different subcomponents operate. The incoming pulse record is passed to two submodules. A detection module combined with a timing one finds the horizontal position of the peak of a pulse. A tunable shaping filter modified the incoming pulses to reduce noise in the pulse height measurement. A sampler uses the timing information to measure an average peak height from the shaped pulse. Different signal filters require a different amount of clock cycles to process a single sample, so they introduce a different delay to the signal. The timing filter must be synchronized to the shaping filter. This is achieved in a synchronization pipeline that delays the faster timing filter by an appropriate number of clock cycles.

3.1 PULSE DETECTION AND TIMING MODULE

Pulse detection is done with the use of a zero crossing detector applied to a digital bandpass filter in the form of a averaging derivative.

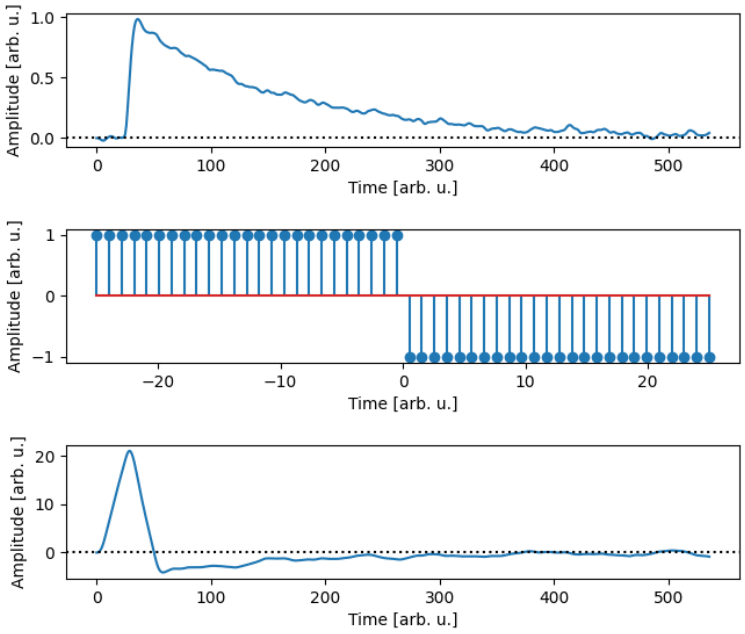

FIGURE 5: BOXCAR FILTER ON A SIMULATED UNIPOLAR EXPONENTIAL PULSE

The incoming pulses are convolved with a boxcar filter. The result of such operation on an exponential pulse, as well as an example boxcar transfer function are shown in Figure 5. A unipolar pulse is transformed into a bipolar signal. The produced signal is a smoothed derivative, so the point at the x-axis is crossed indicates the peak of a pulse. The averaging action present in the boxcar filter causes some delay, so for proper timing it must be accounted for in the timing module, that controls the sampling start.

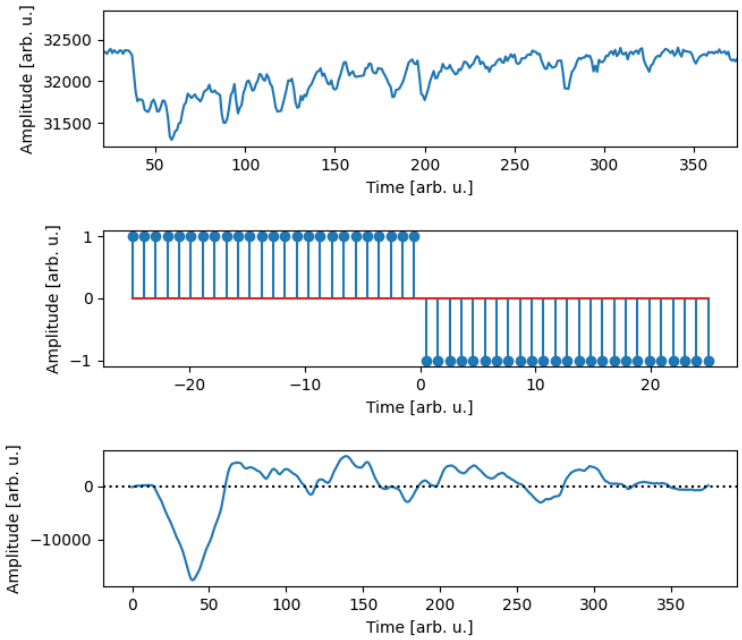

FIGURE 6: BOXCAR FILTER ON A REAL UNIPOLAR EXPONENTIAL PULSE OBTAINED FROM A RADIOACTIVE SOURCE

The boxcar filter is a versatile solution due to its relative simplicity. Figure 6 shows a boxcar filter being applied to a real exponential pulse obtained during tests with a radioactive ^{137}Cs sample. The zero crossing point after the primary signal peak is preserved, however the real signal is substantially less monotonic than simulated exponential pulses. To avoid false triggers, the Pulse Detection module applies thresholding. A threshold level is set using the software package and transferred to the FPGA without a need for reprogramming. When this value is crossed on the boxcar-filtered signal, the detection module begins scanning for a zero crossing. The timing data is sent to the Pulse Sampler and the module awaits another threshold crossing as shown in Figure 7.

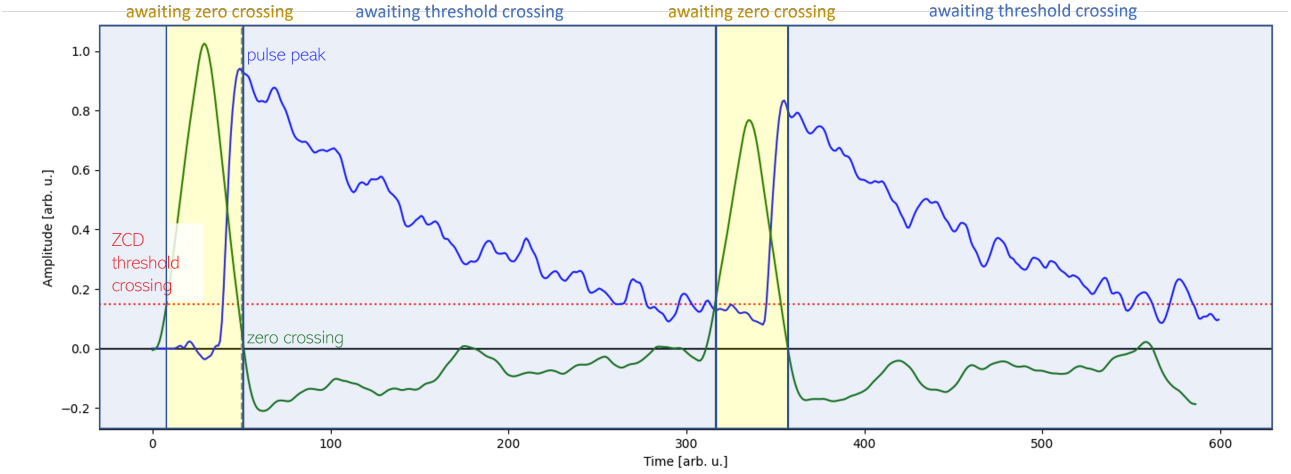

FIGURE 7: PULSE DETECTION THRESHOLDING

3.2 PULSE SHAPING MODULE

A trapezoidal filter is currently used to shape incoming pulses. This type of filter offers a signal to noise ratio second only to a cusp filter. Generally it is not computationally expensive, however it introduces a large gain, that must be attenuated at intermediate points to remove the possibility of an integer overflow occuring in calculations. Figure 8 shows a block schematic of a trapezoid filter. Three points which introduce a substantial amount of gain are highlighted. The gain is dependent on the parameters of the filters. Removing this gain requires division. Dividers can

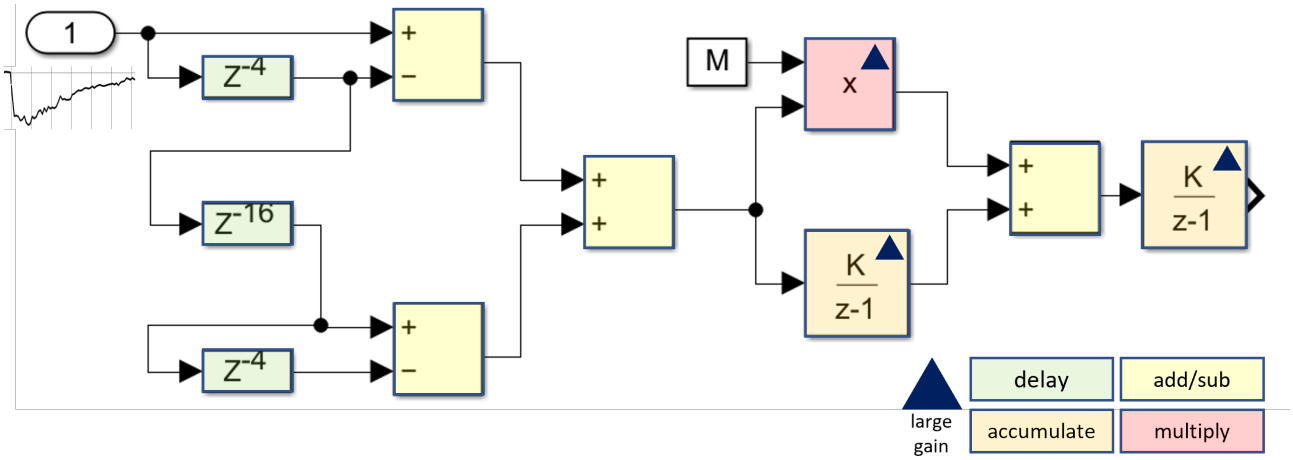

FIGURE 8: TRAPEZOID FILTER BLOCK SCHEME

be implemented in firmware, however they result in complex circuitry, that in turn introduces substantial delay to the signals (20-30 clock cycles). This increases the complexity of other components as they now have to include longer shit registers for synchornization. Where possible dividers were replaced with arithmetic shifters.

3.3 PULSE SAMPLER

Pulses shaped with the trapezoid filter form a flattop, that linearly corresponds to the original pulse height. Due to noise in the system, sampling just the signal peaks would lead to subpar spectrum resolution. The trapezoidal filter performs some averaging in accumulation stages, however the flattop region is never ideally horizontal. Multiple samples are taken at the flattop to improve the result of pulse height measurement. As shown in Figure 9, the pulse sampler operates in states. This acts as a basic form of pile-up rejection. A timing trigger from the Pulse Detection Module causes the sampler to come into active state. After watching the rising edge, upon reaching the flattop region, an accumulator starts adding up samples. The duration of a flattop region is constant and defined by the trapezoid filter parameters, so the necessary division can be performed efficiently, with a reliance on bit shifting or lookup tables.

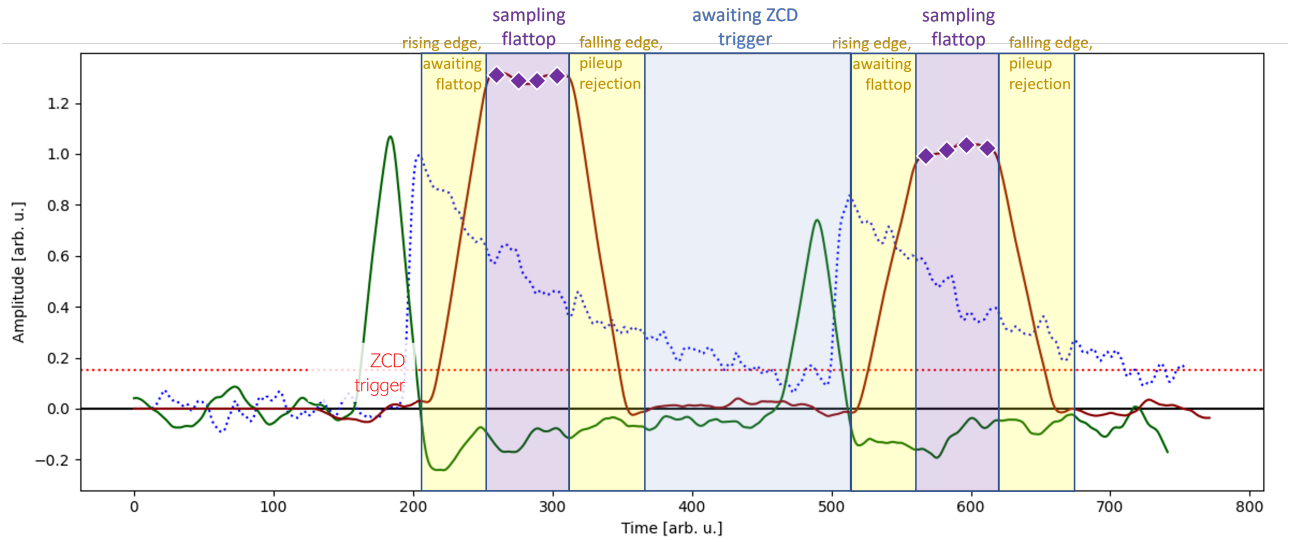

FIGURE 9: TRAPEZOID FILTER BLOCK SCHEME

3.4 SPECTRUM STORAGE AND TRANSFER

The measured pulse heights are binned into up to 2^{14} spectrum bins. Without reprogramming the device the number of bins can be changed into any power of 2 lower than the maximum value. The measured pulse height is initially stored as a 16-bit unsigned integer. The count of pulses in each bin is stored in an individually addressable dual port block RAM consiting of 32-bit words. To translate the pulse height into a spectrum bin, and thus the RAM address, N most significant bits are taken from the pulse height, where N corresponds to the logarithm of the number of bins. This is a very efficient design that does not require divisors nor an array of comparataros, that would otherwise be necessary for binning.

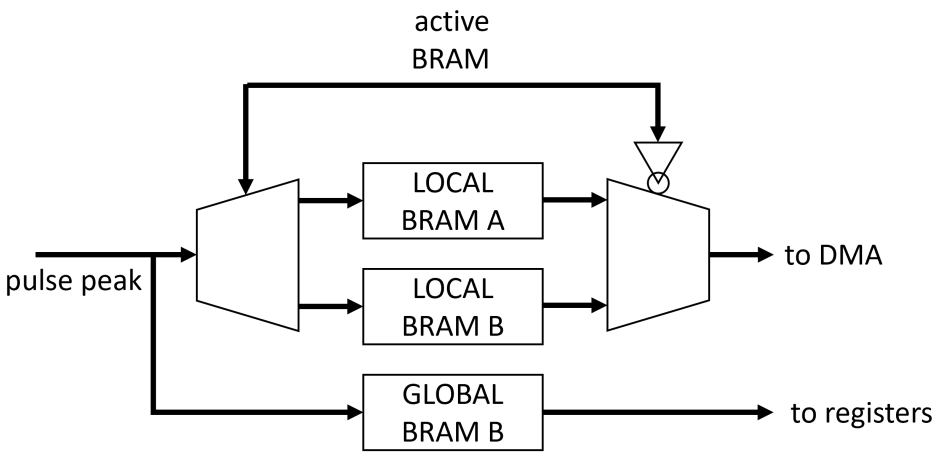

FIGURE 10: LOCAL AND GLOBAL SPECTRUM MEMORY

As shown in Figure 10, a total of 3 block RAMs are present in the design. One, so called global BRAM, is entirely controlled through software. Appropriate bins are incremented as long as a flag in the FPGA registers is not unset. This acts primarily as a debuggin tool or in cases where a spectrum of an entire acquisition is necessary. Spectra are transferred from this BRAM with the use of user registers, so data should be read only when the board is not in an active acquisition.

The other two BRAMs build windowed spectra. The duration of the windows is configurable through the software package. In an example case of a 100 ms window, local BRAM A will be written to for 100 ms, after which time the counting will immedietaly be taken over by local BRAM B. During the next 100 ms window, local BRAM A will transfer its contents over the DMA channel and reset all bin counts back to 0. This happens in approximately 25000 clock cycles, which translates to around 100 μs of down time. With the other local BRAM immedietaly taking over the counting, this down time does not cause missed events, as the active spectrum memory alternates with every window duration.

4 USING CUSTOM FIRMWARE

4.1 QADQSCOPE

The digitizer board, as well as the spectroscope firmware are currently controlled from the same software application called QADQScope. The application allows users to start acquisitions with any configuration available to the board. Figure 11 shows the main window of QADQScope. This

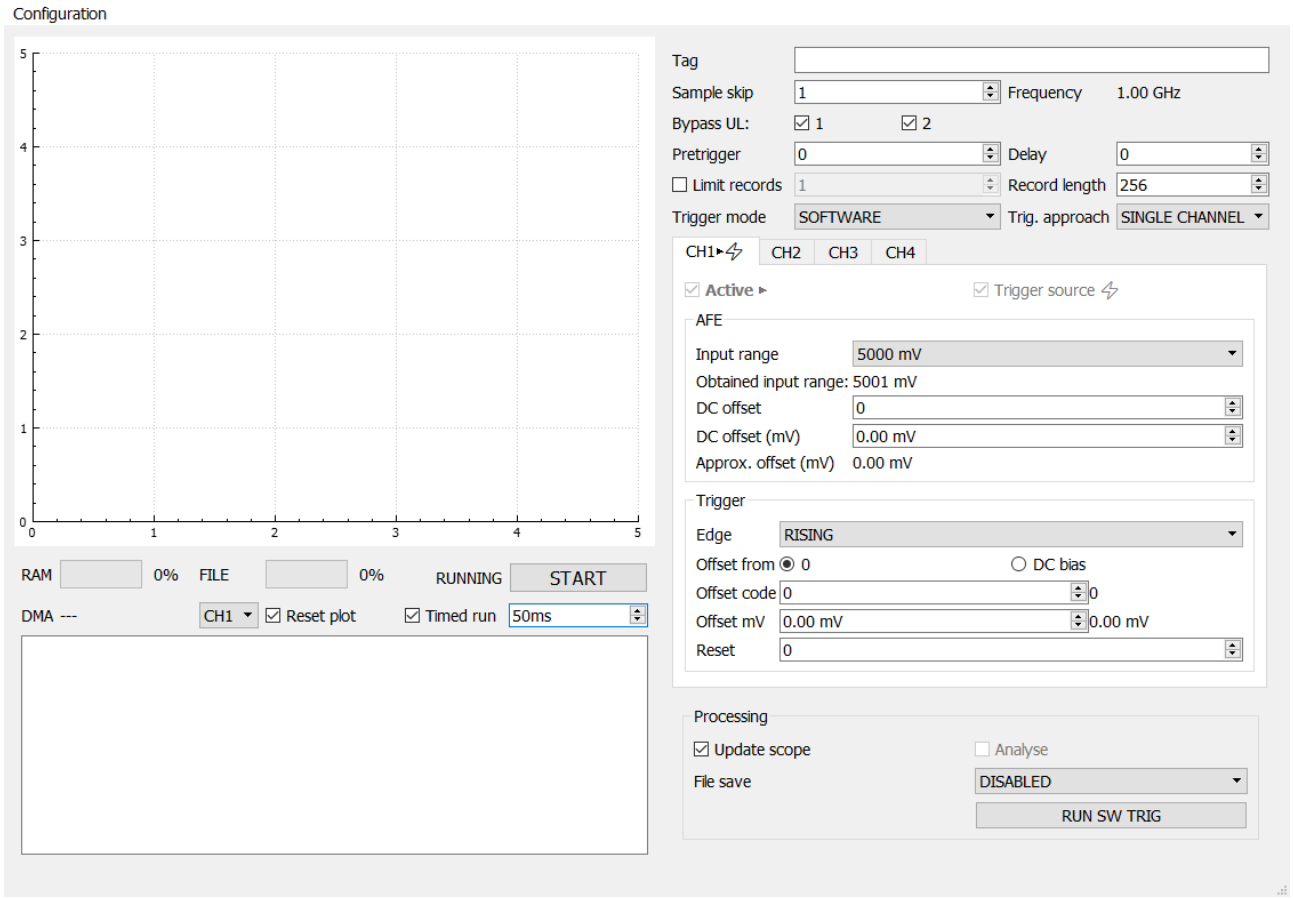

FIGURE 11: MAIN WINDOW OF QADQSCOPE

view is equipped with a plot window capable of displaying incoming records as they arrive to the host PC. Alternatively data can be saved to a hard drive. While it is possible to simultanously save data and display it in real time, the plotting is significantly slower and will cause a bottleneck in terms of processing speed. The primary view lets users change triggering configuration, including record length and horizontal shift. To activate the custom firmware User Logic 2 must not be bypassed.

4.2 SPECTRUM DIALOG

The primary Pulse Height Analyzer are available through a secondary dialog window shown in Figure 12. To enable the spectroscope functionality the appropriate checkbox must be set. This

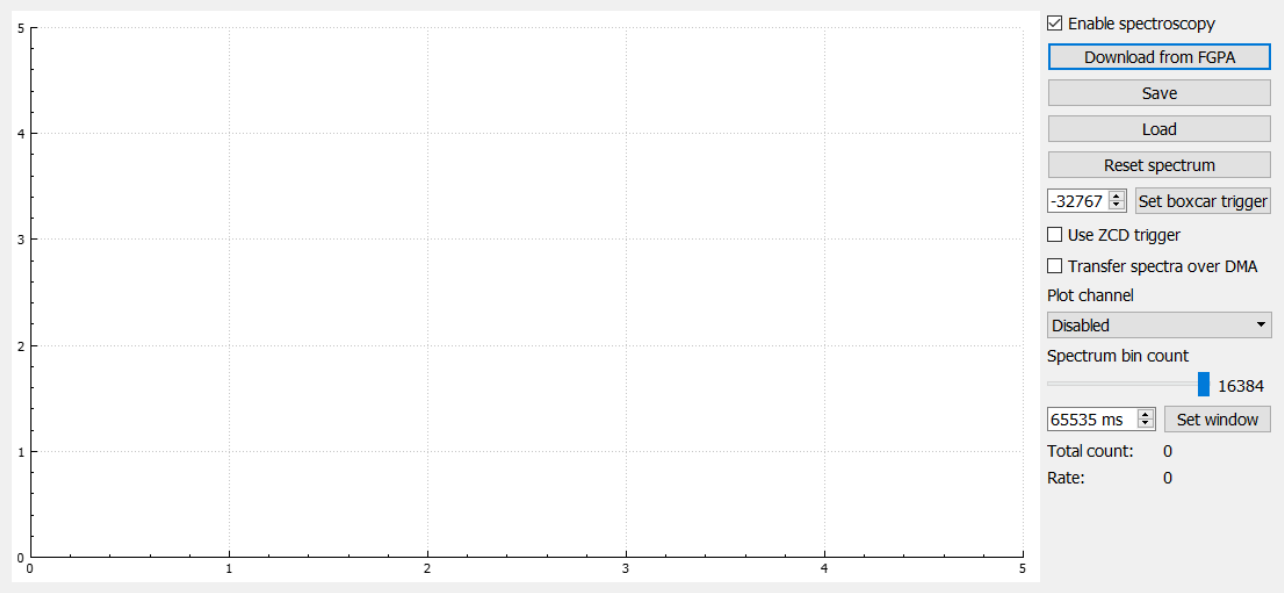

FIGURE 12: SPECTRUM DIALOG OF QADQSCOPE

enables the global spectrum counter. The data from this memory is downloaded at user request when the `Download from FPGA button` is pressed. The `Save` and `Load` buttons allow for spectra to be written to or loaded from .csv files. To reset the counters in the global memory press the `Reset spectrum` button. The boxcar trigger can be set to any value and functions as described in subsection 3.1. The `Transfer spectra over DMA` checkbox enables local memory with a window specified next to the `Set window` button. The transferred windowed spectra can be displayed in real time on the plot available in this dialog window. Changing the bin count is achieved via the slider. After every change to the bin count it is necessary to reset the global spectrum. Otherwise the counts will not be reset automatically and the new data will be superposed with the previously existing one.