Parameterization of Pulses from Electromagnetic Calorimeter Using ADC and FPGA

P. Rajashekar Reddy, Assistant Professor
CVR College of Engineering College, ECE, Hyderabad, India
Email: raju.sheker@gmail.com

Abstract— **The main focus of this paper is data recording and processing. It provides a calibration method for pulses processed by ADCs. It uses some existing filters to achieve the accurate parameters of pulses, and existing concept to do the comparison among filters. This design method with a finite state machine can be realized easily by FPGA. The high precision and the ability of real time application were validated. The FPGA implementation scheme is also discussed in the paper.**

Index Terms— **Particle detector, EMC, FPGA, Pulse parameters.**

I. Introduction

In this paper, some generators will be used to simulate the pulses from Electromagnetic Calorimeter (EMC) detector. The PANDA experiment requires a multi-purpose detector with an important component of a high resolution Electromagnetic Calorimeter (EMC) which is applied to measure the energy of particles. The PANDA detector consists of a target spectrometer (TS) and a forward spectrometer (FS). Those two spectrometers allow the detector to cover a 4π solid angle and each of them employs an EMC.

The paper is based on the development of FPGAs for pulse detection and feature extraction. During pulse signal transmission from the EMC to the FPGA, noise is added to the signal and the signal to noise (S/N) ratio is decreasing. It adversely affects the results of the experiment. One of the goals of this paper is the development of digital data filtration, to achieve a 13-bit amplitude dynamic range of the system (8000:1) for real physical signals, while the raw S/N ratio of an ADC/FPGA system is of the order of 11-bit. Additionally the aimed time resolution for the detected pulses should be of the order of 1 ns or less, while the signal rise-time after amplification and shaping amounts from 20 to 200 ns and the sampling frequency is 80MHz. Therefore, the other goal is to achieve the robustness signal parameterization algorithms which are applicable for the whole range of signal amplitudes.

II. Hardware and Software

Two of the main hardware units used in this work, an Analog to Digital Converter (ADC) and an FPGA, are placed on the same board. The ADC system is used to detect the pulses from analog pulses. The hardware structure needs a high performance analog to digital converter. The requirements are as follows.

A.Amplitude Resolution: Since the range of energy deposited in individual PbWO4 crystals is from 1 MeV up to 8 GeV, a 14-bit ADC is used.

B.Time Resolution: The light pulses created by the crystals have about 20 ns signal rise-time, and hence the ADC should provide 160 MHz with 3 samples on leading edge. But for light pulses which have the 200 ns integrated signal rise-time, the ADC should provide 25 MHz with 5 samples on leading edge.

C.Processing Power: Processing should be flexible and re-programmable with a raw data buffer of 5-10us for each channel. It can also do feature extraction and have external configuration capability.

D. Inside The PANDA Detector: The maximum power dissipation is not fully established but efforts to minimize power consumption have been taken. It is better to have lower sampling frequency and minimize processing power redundancy.

According to the requirements above, a LTC-2175-14 ADC by Linear Technology is used. It is a 4-channel and 14-bit ADC with 125Msps sample rate and 127mW power consumption. There are two FPGAs designed by XILINX® XC5VLX50T, XC3S4000. In this development, a XC5VLX50T FPGA is used. The most important reason for not choosing a XC3S4000 FPGA is that it needs additional components for data de-serialization.

The XILINX® design tools including the ISE® Design Suite and ChipScope Pro™ tools are used in this development. The ISE® Design Suite supports both VHDL (VHSIC Hardware Description Language) and Verilog HDL (Hardware Description Language) for code writing. VHDL is used here.

III. System Implementation Structure

Two main parts are needed here, which are the FPGA programming and the implementation of the ChipScope™ Pro tool. The FPGA is in the core of the work and is responsible for data de-serialization and analysis. The whole structure is shown in Figure 1. In the FPGA part, there are 5 modules. They are Digital Clock Manager, Calibration, De-serialization, Filtration and Parameterization, RAM and Histogram.

The Digital Clock Manager module is in charge of clocks. It takes a 125MHz clock from an LMK0300 clock conditioner as input. Then, it will generate new clocks

DOI: 10.32377/cvrjst0909

with other frequencies such as 200 MHz and 500MHz to drive other modules by using internal phase-locked-loop (PLL) schemes.

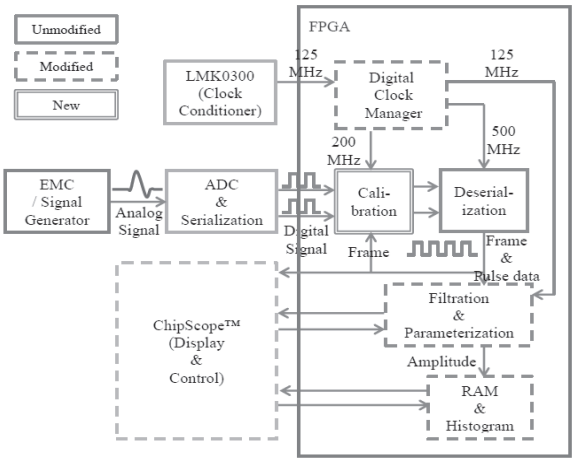

Figure 1: System Implementation Structure

The Calibration module is used to delay the digital signals to get better sampling times in the FPGA. It is controlled by the de-serialized frame signal. The signals which are converted from ADCs are serial signals. The de-serialization module is used to do de-serialization of those serialized signals. Originally, the signals from ADCs include frames and data are sent bit by bit which is synchronized by a transmission clock. This module will also gather bits into single frame and data. The Filtration and Parameterization module contains different filters, for instance, Moving Window De-convolution (MWD), Finite Impulse Response (FIR) or Moving Average (MA) which are used to reduce signal noise. This module is also responsible for calculating the basic parameters for the pulses including pulse width, rise and fall time, and amplitude and so on. Histogram is used to check the parameters such as amplitude and judge if the filters or algorithms suit these pulses. A RAM is implemented because it needs to store pulse data for statistical purposes.

IV. Signal Pre-Process

ADCs (Analog-to-Digital converters) convert analog signals to digital signals after sampling. In order to transfer the results to an FPGA, we use a serial protocol. The FPGA also does sampling in order to get the digital signals. However, the data received might be incorrect and they also need to be de-serialized. In order to receive the correct data, an IDELAY (Input Delay) primitive and a bit slip circuitry in ISERDES (Input Serializer or De-serializer) is used. The LTC-2175-14 ADC made by LINEAR Technology [1] is used here. It has four channels sampling concurrently. The data are transferred using serial LVDS (Low Voltage Differential Signaling) in each ADC channel. The serial LVDS outputs can be 1 or 2 bits per line. Here a 2-Lane output mode is used. Figure 2 shows that an analog signal is transferred to 2 of 8 bits digital signal (OUT#A and OUT#B) with a frame signal (FR). ENC (Encode Input) is the sampling clock which is 125 MHz and DCO (Data Clock Output) is the output clock which is 500MHz. The sample rate is 125Msps (Million Samples per Second) and each sample implies 8 bits per lane. Therefore, there are 1Gbit (125M×8 bit) transferred per second per lane.

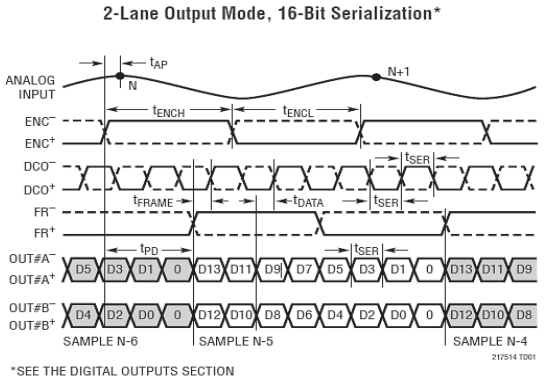

Figure 2: Transmissions In One Of The Channels In One ADC

As shown in Figure 2, after a propagation delay (t_{PD}), a new measurement is made and one serial data bit period (t_{SER}) is half of a DCO period. One serial data bit will be transferred in the middle of the rising and falling edge of DCO. It takes 8 t_{SER} and 4 DCO to transfer 8 bits per lane and 16 bits per channel of the measurement. Each transition of the DCO clock indicates a change in data, commonly called Double Data Rate (DDR) protocol.

A) Serial Data Received In The FPGA

The FPGA will receive the frame (FR in Figure 2) and the output data (OUT#A and OUT#B in Figure 2) from the ADC. The receiving frequency is identical to the output clock of the ADC (DCO in Figure3) which is 500MHz. It is also shown in Figure 2, that data from an ADC is 16-bit serialized and the correct order is OUT#A: D13, D11… D1, 0; OUT#B: D12, D10 … D0, 0. Basically the FPGA should receive the same data in the correct order. However, the FPGA can receive the wrong data since the clock for FPGA to receive data does not synchronize with the clock for ADC to send data.

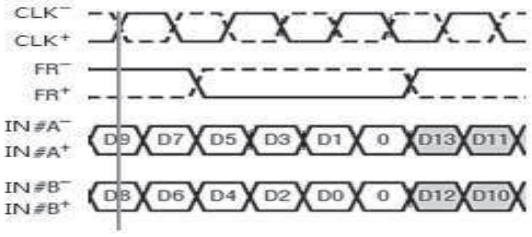

Figure 3: Serial Data Received In The FPGA.

As shown in Figure 3, the FPGA starts to receive the data at D9 and D8 instead of D13 and D12, the FPGA receives the data in incorrect order.

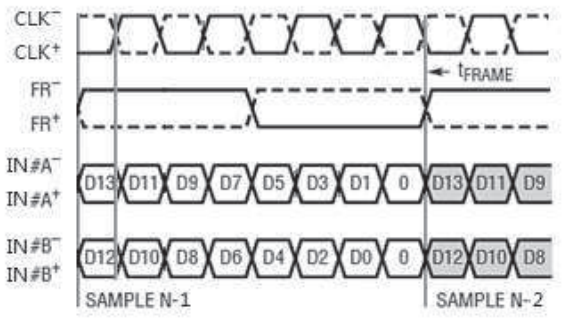

Figure 4: Serial Data Received In The FPGA

Moreover, within the FPGA, the clock does not synchronize with the frame and the receiving data either. The FPGA can start to receive the data on the boundary between two data as show in Figure 4.

B) IDELAY And IDELAYCTRL

In order to solve the problem of the sampling time, the IDELAY (Input Delay element) and the IDELAYCTRL (IDELAY tap delay value Control) primitives from Virtex®-5 will be applied capturing the frame and bits from two-lanes correctly before de-serializing using ISERDES. The IDELAY primitive is a digitally controlled analog delay line as shown in Figure 5.

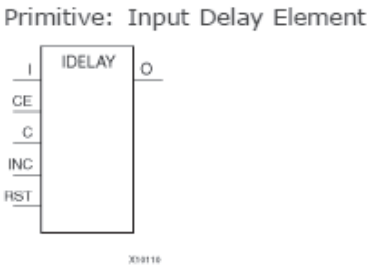

Figure 5: IDELAY primitive [2]

C) Data De-serialization

The outputs of the ADCs are serialized data and the FPGA cannot directly use these data. Therefore, before analyzing the data, an input serial-to-parallel data converter should be employed in doing de-serialization of the frame and the data that come from each ADC.

D) Checksum & Aligning

The FPGA can start to receive data at any time, and it causes two problems as shown in Figures 3 and 4. In order to solve the first problem, the de-serialized frame (FR in Figure 3 is serialized frame) is used as a checksum. The checksum function returns a value to determine if the bit slip function should be performed. If this frame which is still 8 bits after de-serialization is equal to "11110000", the checksum function returns 0, and it means the bit slip function will not be enabled. Otherwise, the checksum function returns 1 and enables the bit slip function to align the bits. The frequency of the clock for the checksum function is 125 MHz, because the decision should be made for every 8-bits frame. But the checksum function checks the frame every 4 clocks, in order to give a time for the bit slip function in ISERDES to do the aligning. In Figure 3 as an example, the FR is "11000011", but not "11110000". In this case, the checksum function returns 1 in the first clock. Then, the bit slip function will be performed to do the aligning in the next three clock times. Checking and aligning one to be repeated until the checksum function returns 0.

E) Calibration

Figure 4 only shows the received data from one of the channels of the ADC. Here each channel has two lanes, each ADC has 4 channels and there are 4 ADCs, totaling 32 lanes. Therefore, all the data which are received from the ADCs should be as shown in Figure 6. If the FPGA starts to receive data at position 2 in Figure 6, the checksum function can return 0 and the aligning will be enabled, but the received data can still be incorrect since it is close to the noise. This is the second problem.

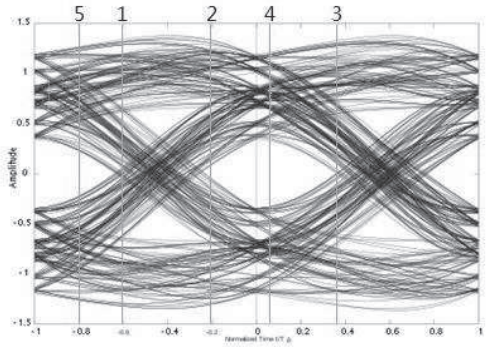

Figure 6: Worst Case Of Sampling

The theoretical maximum eye opening is 1ns per lane. All lines are sample-clocked with one clock signal and the goal is to sample all data lines in the middle of the eye (position 4 in Figure 6), where the signal has the highest distance from noise. In order to reach the goal, a state machine is introduced, since the checksum and aligning are not enough.

F) Filtering

A pulse must be filtered in order to significantly reduce the noise. There are a lot of filters in the world applied in this field. But we would like to compare them in such a way that we know which one is better to use in our case. Finite Impulse Response (FIR) Filter, Moving Average (MA) and Moving Window Deconvolution (MWD) is used here for comparision. In order to measure the precision of amplitude estimations from different filters, a normal distribution table or a histogram of amplitudes will be created and analyzed. The histogram is a graphical representation of the distribution of data. It is an estimate of the probability distribution of a continuous variable. Histograms are used to plot the density of data and get the probability density function of the underlying variable.

DOI: 10.32377/cvrjst0909

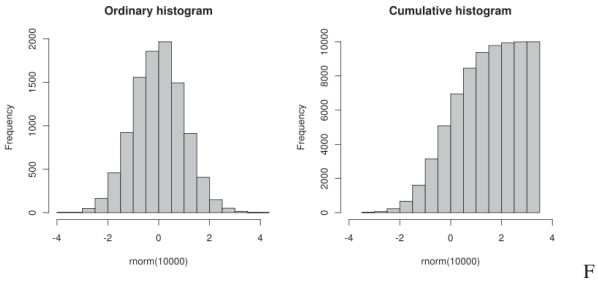

Figure 7: Examples of histogram [3]

For example in Figure 7, after running for a number of times, a graph where the frequencies are located around 0 is the one similar to a normal distribution. The more samples the testing has, the more precise the distribution will be. Here the estimation precision of amplitudes from different filters will be measured. The goal is to create and analyze the normal distribution table or histogram of amplitudes from different filters.

V. Parameterization

The parameterization is implemented after filtering the pulse. The general parameters of the pulse include pulse amplitude, pulse width, pulse arrival time etc. The shape of the pulse is shown in Figure 8 with rise and fall time, The amplitude is not so stable due to the influence of noise. The pulse parameter definitions are provided as follows [4]. The Pulse Amplitude (PA in Figure 8) is the difference between the maximum value and the minimum value of the pulse. Pulse Rise Time (PRT in Figure 8) is the interval between the 10% and 90% amplitude points on the leading edge. The Pulse Fall Time (PFT in Figure 8) is the interval between the 10% and 90% amplitude points on the trailing edge. The Pulse Width (PW in Figure 8) is the interval between leading and trailing edge medians. The specified and displayed value is that obtained with fastest edges, essentially equal to the interval from the start of the leading edge to the start of the trailing edge. The Pulse Integral (PI in Figure 8) is the sum of the differences between the value of pulse and the minimum value of pulse during the sampling time.

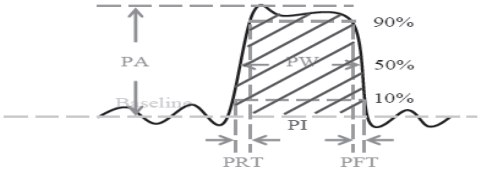

Figure 8: Pulse Waveform And General Parameter Definitions

A) Baseline

Baseline is an important element for calculating the pulse parameters. The baseline will shift over time, and therefore, in order to calculate the most accurate pulse parameters, the baseline should be calculated continuously. However, it is difficult to find the baseline for the pulse because there is too much noise around the baseline rather than a fixed value. Moreover, as each sample is read into the FPGA, the first step is to determine if the sample is part of an event or a sample of the baseline. If the sample is part of an event pulse, it should not be included in the baseline calculation. To exclude samples of an event pulse, a baseline window [5] giving a range of the baseline with upper and lower limits will be calculated instead of a single baseline.

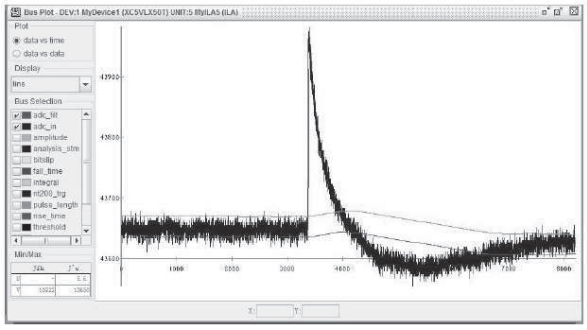

Figure 9: Baseline Window

If a sample is inside the baseline window, it is determined to be part of the baseline; otherwise it is assumed to be part of an event pulse. The baseline window keeps track of how many samples have been inside and how many have been outside the window. Initially, the value of the upper baseline is 2 ADC least significant bits (LSB) and the lower baseline is 0 ACD LSB. If the following 64 samples are larger than the value of the upper baseline, both the upper and lower limits of baseline window will be increased by one ADC LSB. If the following 64 samples are smaller than the value of the lower baseline, both the upper and lower limits of the baseline window will be decreased by one ACD LSB. Or if 512 samples have been inside, the window is contracted by one ADC LSB. Likewise, if there are 64 samples outside, the baseline window is expanded by one ADC LSB. The upper and lower limits of the baseline window will be increased or decreased a lot in the beginning, because the baseline window is far away from the pulse. Once the samples get inside the baseline window, contraction or expansion will happen a lot. In order to simplify the control, the three counters (above, below or inside the baseline window) only reset when 64 or 512 are reached respectively. The window expands faster than it contracts because there are more baseline values than pulse values. In this data set, the ratio is about 4:1. Figure 9 illustrates an example of a baseline window (the upper limit of baseline window is in light green and lower limit of baseline window is in dark green).

B) Threshold

A threshold is used to separate the pulse from the baseline. It is almost the same as the upper limit of the baseline window. The difference between them is that after the pulse rises and crosses the upper limit of the baseline window, the upper limit will increase until the pulse goes down and crosses again, but the threshold should be kept at the same value until the pulse and upper baseline cross again. This case gives a better and more accurate value to calculate the amplitude. The amplitude will be calculated

by the maximum value of the pulse and the threshold (A in Figure 10) instead of the maximum value of pulse and the upper limit of the baseline window (A' in Figure 10).

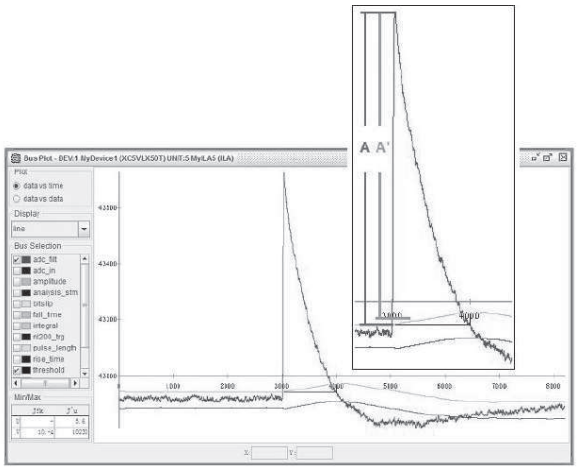

Figure 10: Threshold

Figure 11 shows that the length between two intersection points created by a pulse and the threshold (TH1) are the arrival time of the pulse rising edge (t1) and the arrival time of the pulse falling edge (t2). All of the parameters will be calculated during the pulse width PW'.

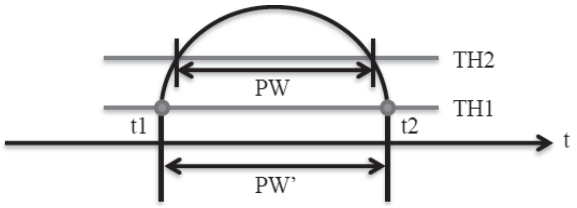

Figure 11: Pulse Width

The pulse amplitude (PA) is the most important parameter to describe the pulse waveform. The measurement of PA directly affects the measurement of other parameters. For instance, in order to measure the Pulse Rise Time and the Pulse Fall Time, the PA should be measured first. Here we start to search for the PA from point t1 as shown in Figure 11. For each clock time, a variable whose initial value is 0 will be compared with the current value of the amplitude (the difference between pulse and threshold). The variable will be replaced by the current amplitude if it is smaller than current amplitude, otherwise, there will be no change. When t2 is achieved, the search will stop and the variable will be returned as the PA. The calculation of pulse rise time (PRT) and pulse fall time (PFT) are similar to the PA calculation. If the current amplitude is equal to the variable, the PRT will be increased by one. Otherwise, if current amplitude is smaller than the variable, the PFT will be increased by one. There are no changes when they are equal.A rough value of the pulse width (PW' in Figure 11) can easily be measured by starting up a counter at t1 and increasing it by 1 each clock cycle. The return value of the counter at t2 is PW'. However, the real PW is the time interval between two points which are the intersection between the pulse and TH2. Threshold TH2 is half of the pulse amplitude. Therefore, it is difficult to define TH2 before the PA has been measured. There are two methods to calculate PW. One is, saving all of the values with time during PW', then searching the memory and calculating after the amplitude has been calculated. Another is, using the amplitude of previous pulse as reference to define TH2, and then doing a similar calculation of PW'. The first method gives a more accurate value but wastes memory space. The second method is used because the input pulses will not be changed for a period of time and the measurements are similar during this time. For each clock cycle during PW', the current values (amplitudes) will be accumulated. The sum of those values is the pulse Integral (PI).

VI. Testing

The histograms obtained by using a Xilinx in-chip logic analyzer (ChipScope Pro™) for later analysis, MATLAB, a numerical computation, visualization, and programming environment will be used to do the comparison among histograms. This is hard to do in real time.. The Full Width at Half Maximum (FWHM) value as criteria will be calculated from the histograms by using MATLAB.

A) Full Width at Half Maximum

Criteria should be set for measuring the accuracy of amplitudes calculated from different filters. In this paper, the full width at half maximum (FWHM) [6] which is a parameter given by the difference between the two extreme values of the independent variable at which the function reaches half its maximum value will be used as the criterion. The smaller the FWHM is, the better the filter is. The best case is when FWHM equal to 1. The measurement of FWHM is similar as PW as shown in Figure 9, but the difference is that FWHM is calculated in post processing and PW is calculated in real time. The value x in Figure 12 represents amplitude. The function f(x) represents the frequency distribution of achieved amplitudes within 65525 times of the experiment. Each function will be traversed twice here. The maximum value of time for the first time achieved by keeping the larger value during traversal. The second time, the FWHM is calculated by starting a counter when the value is larger than half of the maximum value and finishing it when the value is smaller than half of the maximum value

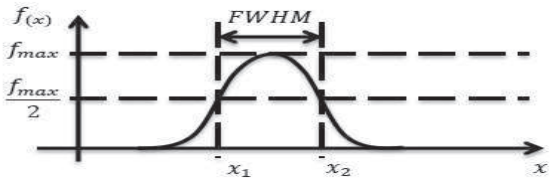

Figure 12: Full Width at Half Maximum

VII Analysis

The comparison among different filters is based on the FWHM values which are achieved from 8 different pulses,

DOI: 10.32377/cvrjst0909

both single-humped pulse and double-humped pulse with four different amplitudes (2mV, 10mV, 50mV and 500mV). Each pulse with different filters will be tested 20 times and the average values of the FWHM will be calculated for analysis as shown in Figure 13(a) and (b). Figure 13(a) illustrates that FWHM values of MA are much lower than MWD, which means MA is much better and more stable than MWD here.

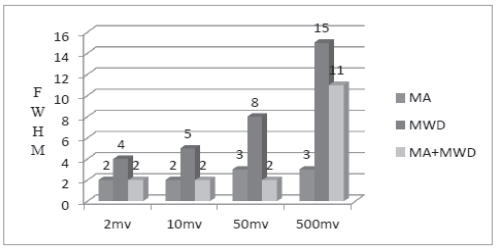

Figure 13 (a): FWHM values for single-humped pulses with different filters

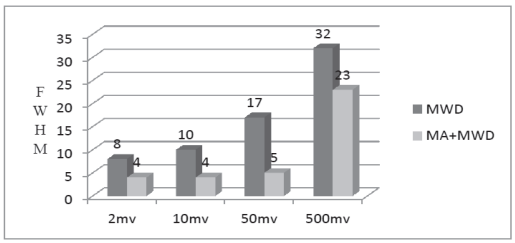

Figure 13(b): FWHM values for double-humped pulses with different filters

The FWHM values of the combination of MA and MWD is among the three. The stability of MA & MWD is better than MWD but worse than MA. As shown in both Figure 13(a) and (b), with the increase of amplitudes, the values of FWHM become larger and larger, and all the filters become more and more volatile. A double-humped pulse means two values of amplitude are received from each test. The FWHM values of it should be double the value of single-humped pulse with one value of amplitude. From the comparison between Figures 13(a) and (b), it is easy to see that the values of FWHM in (b) are almost double of the values of FWHM in (a). Therefore, it means the stabilities of filters will not be influenced by whether they are single-humped pulses or double-humped pulses.

VIII. Results and Conclusion

The investigations provide a calibration method for pulses processed by an ADC. They use some existing filters to achieve the accurate parameters of pulses and existing concept to do a comparison among filters. The FPGA implementation is also discussed. The investigations have a realistic significance in the high energy physics, especially in particle analysis. The main achievements are embodied in the following aspects.

(1) Introduced a calibration method for achieving an accurate digital signal after the analog signal is converted and serialized by ADC and serialization. This design method with a finite state machine can be realized easily by FPGA. The high precision and the ability of real time application were validated.

(2) Combined two existing filters, MWD and MA. MWD filter can avoid signal overlaps and make the measurement of amplitudes of double-humped pulses to become easier. The MA filter reduces the effect of noise on the accuracy of measured parameter and increases the S/N ratio.

(3) Provided measuring methods for different parameters which are also suitable for real time measurements. The most important parameter is the amplitudes of pulses. Accumulated the values of amplitudes and returned histograms with software ChipScope Pro™.

(4) Calculated the FWHM for each histogram and compared different filters with FWHM. All of the back-end processing was done in MATLAB. After multiple testing, the MA filter holds the highest stability, MWD filter holds the lowest, and the combination of them is in between. However, it is difficult for measuring double-humped pulses by only using MA. Therefore, the combination of MA and MWD is the best choice when measuring double-humped pulses, and it is enough use MA when measuring single-humped pulse waves.

(5) Selected an XC5VLX50T FPGA, and illustrated the structure of a developed board. The whole implementation procedure and part of pseudo codes are also given.

There is room for improvement with regard to the filters. By using both MA and MWD, one can easily achieve parameters in both single-humped pulses and double-humped pulses, but with an increase of the amplitudes of the pulses, the stability of MWD is decreasing a lot, and the stability of the combination of MA and MWD also decreases a lot. Future investigations should focus on finding filters which are more efficient and easy to realize in FPGA. Kalman filtering for example, which is an optimal estimator and a widely applied concept in time series analysis, can be used for future implementation. It is known as linear quadratic estimation and it is an algorithm that produces a statistically optimal estimate of the unknown state of the dynamic system from noisy data taken at discrete real time [7].

References

[1] LTC2175-14, “14-Bit, 125Msps Low Power Quad ADCs” Linear Tech-nology Corporation.

[2] Virtex-5 Libraries Guide for HDL Designs. XILINX®; UG621 [Inter-net]. 2009 Sep 16 [Cited 2014 Apr 21].

[3] File:Cumulative vs normal histogram.svg [Internet]. Wikipedia, the free encyclopedia. [Cited 2014 Apr 21]. IEEE Standard Pulse Terms and Definitions. IEEE Std 194-1977. 1977; 1–23.

[4] Haselman M, Hauck S, Lewellen TK, Miyaoka RS. “FPGA-based pulse parameter discovery for positron emission Tomography” Nuclear Science Symposium Conference Record (NSS/MIC), 2009 IEEE [Internet]. IEEE; 2009 [Cited 2014 Apr 21]. p. 2956–61.

[5] Weisstein EW. “Full Width at Half Maximum rom Wolfram Math-World [Internet]. [Cited 2014 Apr 21].

Chui CK, Chen G. Kalman “Filtering with Real-Time Applications” Springer Science & Business Media; 2008. 241 p.

DOI: 10.32377/cvrjst0909