July 19, 2018

Characterization of Self-Triggering on the PSEC4 Waveform Digitzing ASIC.

John Podczerwinski

Abstract

The PSEC4 is a $5 - 15$GSa/s, 1.5GHz bandwidth, 256 sample waveform digitizing ASIC. This paper describes tests done to characterize the PSEC4's self-triggering feature. These were performed using software that scanned through a variety of reference voltages on each channel, measuring the trigger rates at each point. The results of these tests were then used to characterize noise trigger rates, self-triggering efficiency for pulses similar to those provided by Large Area Pico-second Photo-Detectors (LAPPDs), and the dependence of self-triggering performance on input pulse width, DC offset value, and comparator resistor value. It was found that the second channel of each chip had noise triggers occur over reference voltage ranges much wider than other channels, with widths on the order of $10mV$ as opposed to widths less than 2mV. Moreover, pedestal offsets require that LAPPD shaped pulses be larger than 60mV for at least 4 out of 6 channels on each chip to be included in the self-trigger mask, 40mV for 3 out of 6 channels, and 20mV for 2 out of 6 channels. Trigger region width was found to achieve a maximum at a pedestal voltage of 292mV and found to increase by 20% when pulse width rose from 2.5ns to 25ns. Lastly, reducing the comparator resistor value from 1kΩ to 200Ω was found to increase both self-triggering sensitivity and noise-triggering.

Contents

1 Introduction

1.1 The Large Area Picosecond Photo-Detector (LAPPD)

The LAPPD is a 20cm-squared MCP-based photo-detector capable of providing timing resolution under 10psec for single photons and a spatial resolution of $700\mu m$. It consists of a photo-cathode window, two MCP's, and a set of 30 anode strips for readout.

The large area and high degree of timing and spatial resolution make the LAPPD useful for many applications. In the realm of high energy physics, among other applications, LAPPDs would be useful for track reconstruction in large Cherenkov detectors, and for extracting quark information in collider experiments. The LAPPD is also being used in the development of medical imaging techniques. [1]

1.2 The PSEC4 Readout System

In order to read out fast waveforms, switched capacitor array (SCA) analog memory was developed. SCAs allow for waveforms to be sampled at a high rate, but at the expense of slow readout times. Such technology is well suited for high energy physics, where fast sampling rates allow for precise arrival time measurements, and a bit of dead time is acceptable [2][3].

In order to read out the fast waveforms produced by the LAPPD, new waveform sampling technologies had to be developed. In 2011, Eric Oberla and Hervé Grabas developed the PSEC4 ASIC, an SCA based, 6 channel, 5-15GSa/s, 1.5GHz bandwidth waveform digitizing chip [4]. Oberla also designed a front end board called the ACDC, on which 5 of these PSEC4 chips are mounted, giving 30 channels of PSEC4 per board.

The ACDC board is connected via cat5 cables to the ANNIE Control Card (ACC), designed by Mircea Bogdan at the University of Chicago. The ACC can control up to eight slave boards via ethernet, and serves as a communication point between the computer and the electronics setup. [6]

1.3 Motivation for Testing the Self-Triggering

Due to its relatively short buffer length (between 25ns and 40ns during typical operation), any event of interest will be overwritten unless the PSEC4 receives a trigger signal soon after the event occurs. The self-triggering feature on the PSEC4 chips can provide the fast triggering required, and as such it is of interest for experiments looking to use PSEC4. The purpose of this report is to characterize the self-triggering feature, so that experimenters can use it reliably.

2 Testing Setup

These tests were all performed using an ACC / ACDC DAQ of the sort described in section 1.2. The ACC and ACDC boards used here run on 25MHz clocks. The ACDC

firmware was also set up such that all PSEC4 chips receive a 25MHz write clock (see figure 1).

Two ACDC boards were used for these tests, which are called ACDC10 and ACDC20. ACDC10 has a 40MHz on board oscillator, while ACDC20 has a 25MHz oscillator. This shouldn't make difference though, since the firmware of ACDC20 was adjusted to compensate for the slower clock.

To generate pulses, an AWG5012 Arbitrary Waveform Function Generator was used for all but one of the tests. An AWG7012 was used to perform the width dependence testing, as the AWG5012 wasn't capable of producing 2.5ns wide square pulses.

For all but the width dependence test, a pair of 1650 Tri-Output Power Supplies were used. The negative terminals of these power supplies were connected together. For width dependence testing, both board were powered with a single switching power supply. Noise measurements were taken with both the switching power supply and the 1650 Tri-Output Supplies (which are analog) and the noise in each test was found to be comparable.

The standard ACC firmware and 25MHz ACDC firmware[1] were used to perform these tests. Eric Oberla's acdc-daq software was also used, although with some slight edits made to get it to work with the ACC rather than the old central card (CC). Three functions were added to the software to automate testing.

It should also be noted here that all of these tests were performed with the PSEC4 comparators set to trigger on the falling edge of pulses.

3 How PSEC4 Self-Triggering Works

Each channel of the PSEC4 chip contains a comparator used for generating self-trigger signals, as can be seen in figure 1. The positive comparator input is supplied with signals from the channel's signal input pin. These signals are AC coupled and given a DC offset. This DC offset is based on a programmed value called the *pedestal* (denoted P in the plots). The negative comparator input pin is supplied with a voltage based on a programmed value. This programmed value will be referred to as the *DAC threshold*, denoted t_{DAC} in the plots. For chips located on the ACDC board, the threshold and pedestal voltages are supplied by a Digital to Analog Converter (DAC). The DACs supply pedestal and threshold voltages on a chip-by-chip basis. Ideally, when the signal voltage dips below the voltage on the negative comparator pin, the comparator fires, sending a signal to the FPGA. The FPGA then sends back a trigger signal which stops the PSEC4 from sampling, allowing the waveform to be digitized and read out.

[1]This can be found on the LAPPD-DAQ github page: https://github.com/lappd-daq.

The values P and t_{DAC} are called *programmed* values, or values that the software requests that the DAC's produce. In practice, the values that a DAC will produce may differ by up to a few mV. Below are some sample comparisons between programmed and actual DAC voltages, using $P = 292mV$ and $t_{DAC} = 263mV$, for a programmed difference of $29mV$.

Chip	Measured Pedestal (mV)	Measured DAC Threshold (mV)	Measured Difference (mV)
1	287.2	255.4	28.8
2	288	258.3	30
3	284	259.5	25
4	284.7	255.5	29
5	285.6	257.4	28.2

For the remainder of this report, t_{DAC} and P will always refer to programmed values. In checking the DAC pins, it was found that t_{DAC} and P are related to their measured counterparts t_m and P_m by equations

$$t_m = \alpha_1 * t_{DAC} + \beta_1, P_m = \alpha_2 * P + \beta_2.$$

In a test on one of the chips, α_1 was measured to be 0.96 and α_2 to be 0.96 as well. the slopes for each of these are close enough to 1 that plots made using t_{DAC} and P will not be qualitatively any different than those produced using P_m and t_m. The values for β_1 and β_2 are not known but are not needed, since none of the conclusions depend on knowing their values. In other words $t_{DAC} \approx t_m$ and $P \approx P_m$.

4 Noise Trigger Testing

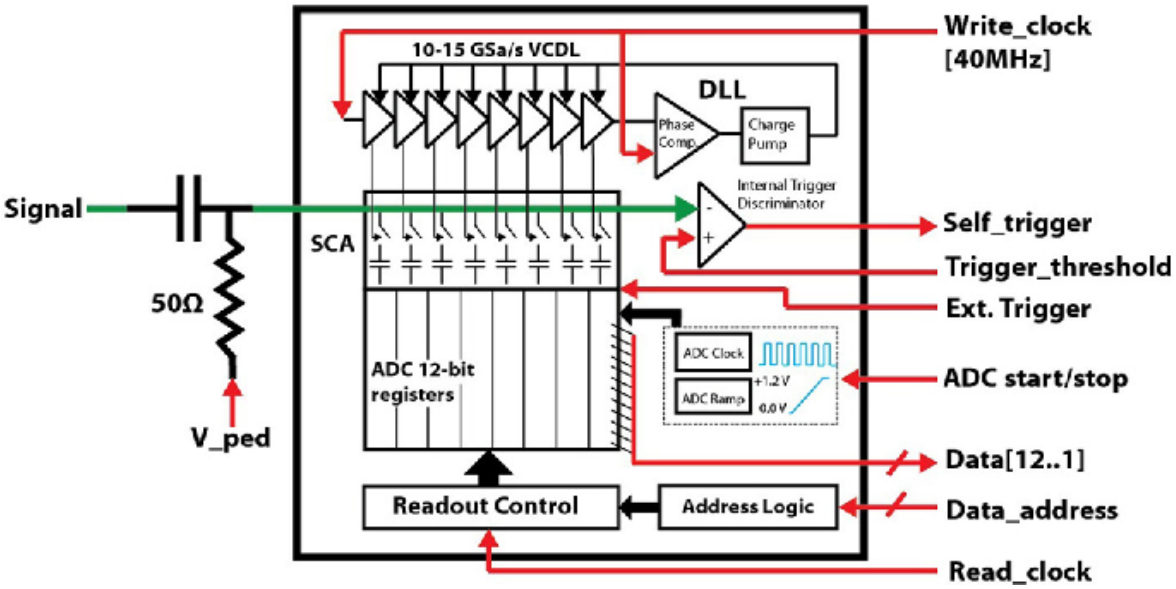

Figure 1: A block diagram showing the architecture of one PSEC4 channel. Courtesy of Eric Oberla.

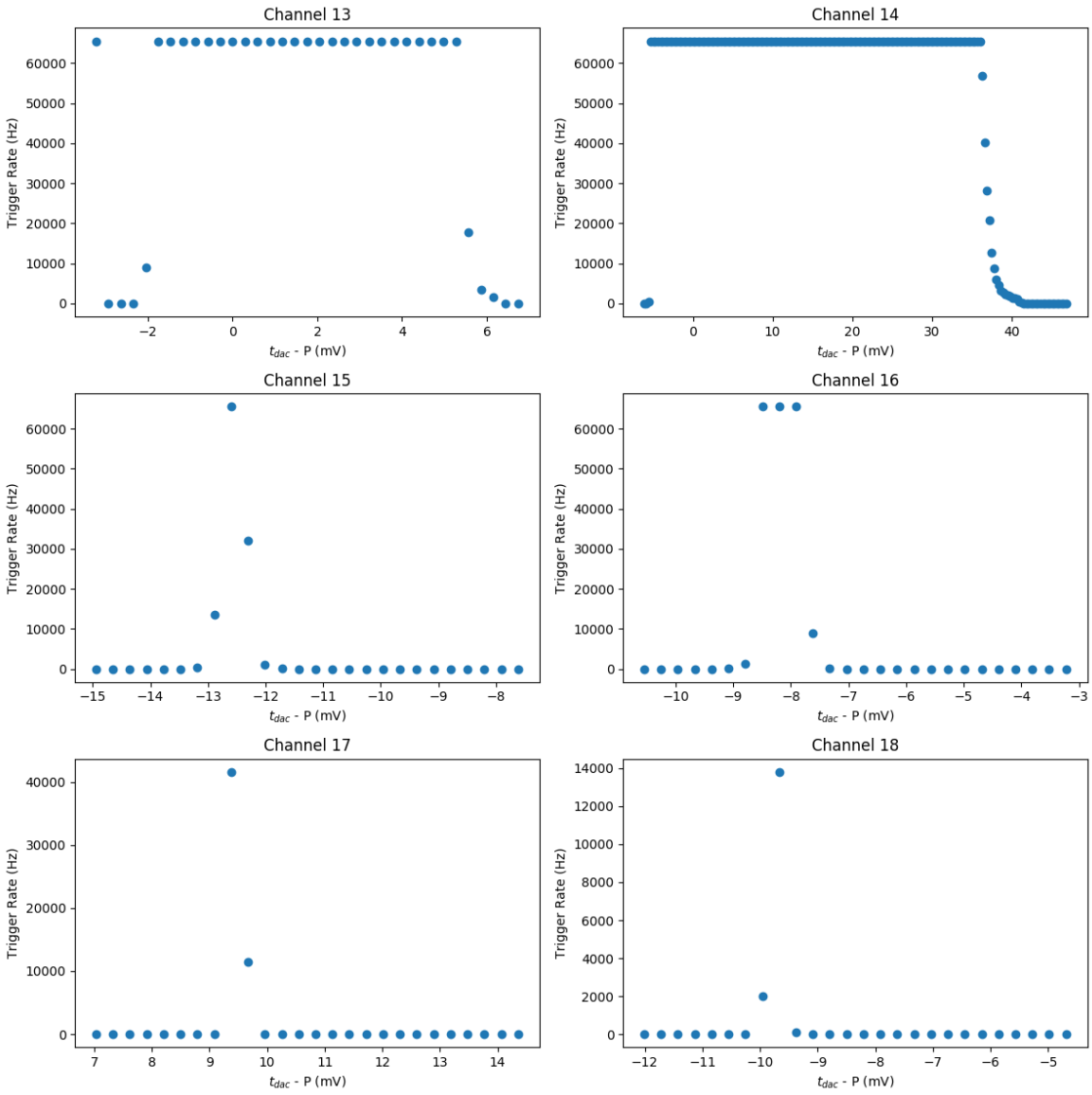

Figure 2: Noise trigger rates in Hz as a function of DAC threshold minus Pedestal for chip 3 on ACDC10. For this test, the pedestal voltage was set to 858mV. The highest reported rate is 65535Hz since the trigger rate is written to a 16 bit register.

In order to characterize the noise trigger rates on the PSEC4 chips, a function was added to Eric Oberla's ACDC-DAQ software. This function keeps the pedestal voltage fixed and scans through a range of DAC threshold voltages in increments of about 1.5mV, measuring trigger rates at each point. When the trigger rate exceeds 10Hz, the program reduces the DAC threshold by 1.75mV and begins scanning in 300μV increments, writing the trigger rate at each point to a file.

Fig. 2 gives some typical plots produced by this test. Note that for each channel, Fig. 2 shows the DAC threshold relative to pedestal at which noise triggering begins. The DAC threshold minus pedestal voltage difference at which noise triggering turns on

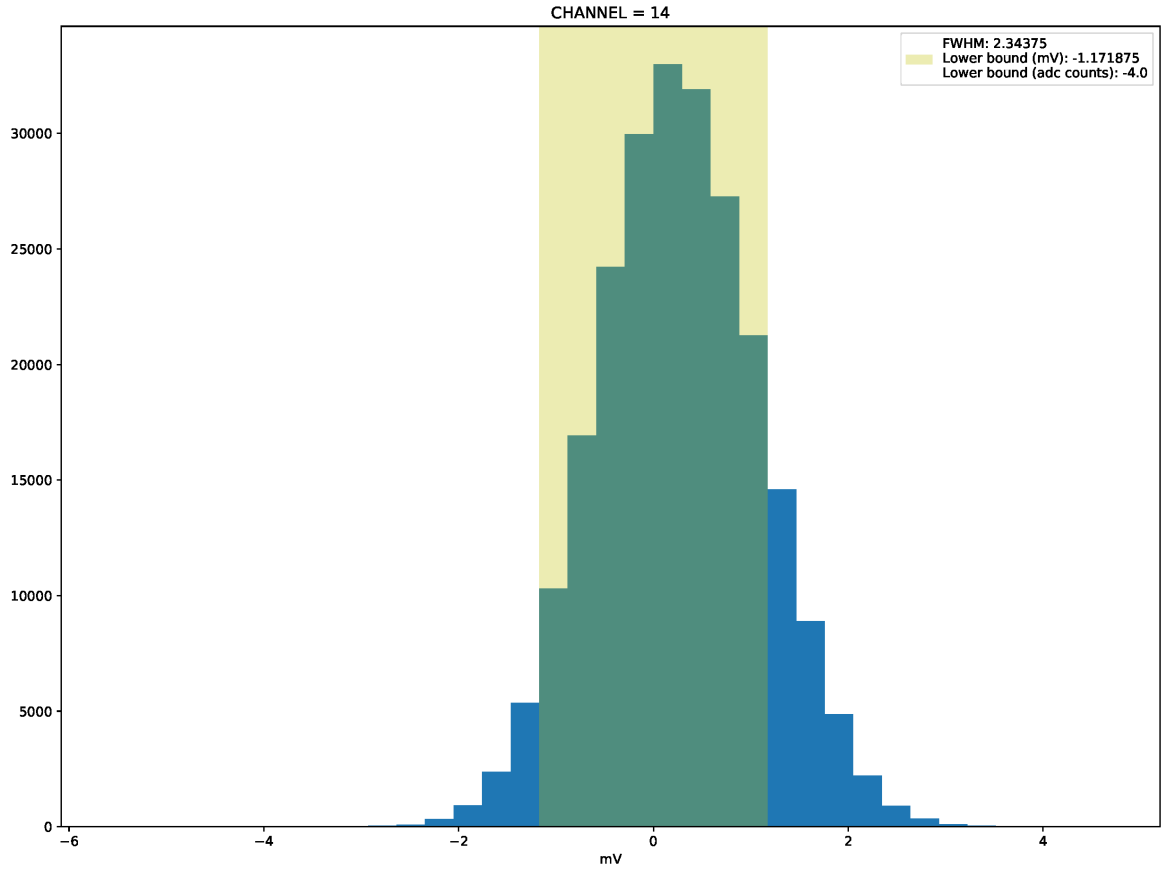

Figure 3: A histogram showing the number of samples with a given noise amplitude taken over 1000 events. The FWHM is reported to be about 2.34mV. Courtesy of Evan Angelico.

varies from as low as about -12.5mV for channel 15 to as high as 9.5mV for channel 17, a range of roughly 20mV. This behavior, that each channel behaves as if it were supplied with a different pedestal value, is observed despite the fact that all channels are given the same voltage by the DAC. This *offset effect* is present and behaves similarly for every batch of PSEC4 chip made, and occurs on both ACDC boards and Eval Cards. This offset effect is believed to be due to manufacturing variations and to be inherent in the comparator components of the PSEC4 chips. In Sec. 5, it will be shown that the offset effect applies to self-triggering on pulses as well.

Another thing to notice in the plots on Fig. 2 are the ranges of threshold values in which each channel of the PSEC4 chip trigger on noise. Channel 13 (the first channel of the chip) triggers over a range of about 8mV. Channel 14 (the second channel on the chip) triggers on noise over a range of roughly 50mV. The rest of the channels on the chip trigger over on noise range much smaller, less than 2mV for each.

It was found that this is a pattern for the PSEC4 chips on the ACDC boards. i.e. the first channel on the chip triggers on noise over a range of a few mV, the second channel over tens of mV, and the others over smaller ranges (<2mV). A quick test was

also performed on an "eval card", which is a different front end board containing only 1 PSEC4 chip. On the eval card, channel 2 also triggered on noise over a wide range of DAC thresholds. The rest of the channels triggered over <2mV, if at all.

Despite the large range of threshold voltages in which noise triggers occur, the signal inputs on channels 13 and 14 did not seem any noisier than any of the other channels. Fig. 3 shows a histogram of noise amplitudes for channel 14. The FWHM of this plot is $\approx 2.34mV$, which of course is much smaller than the tens of mV range that the plots in Fig. 2 might imply. The threshold pins of channels 13 and 14 were checked with an oscilloscope probe and it was found that they were not significantly noisier than the thresholds for any other channel on the board.

5 Triggering on Pulses

To characterize self-triggering on pulses, "turn on curves" were produced. The turn on curves show self-triggering efficiency as a function of DAC threshold voltage relative to pedestal voltage. To perform this test, identical pulses were supplied to all channels on the board via its calibration input, and a program similar to the noise testing code was used. The pedestal voltage was kept fixed and DAC threshold values were scanned over in increments of about 1.5mV, with the trigger rate being measured at each point.

When the trigger rate surpassed 1kHz (supplying pulses at 10kHz), the program would reduce the DAC threshold by 1.75mV and begin a fine-grained threshold scan, incrementing by 300μ V. At each DAC threshold in the fine-grained scan, the program would measure the number of triggers with the calibration input disabled over five 1 second intervals, giving the noise trigger rate at that threshold. It would then do the same but with the calibration input enabled. This fine-grained scan incremented DAC threshold until self-triggering stopped or the noise trigger rate exceeded 200Hz. 200Hz was chosen as the stopping point for the scans as noise-trigger rates typically shoot up to the tens of kHz 300μV or 600μV after that.

Self-triggering efficiency was estimated as

$$Efficiency = \frac{N_{enabled} - N_{disabled}}{t * f}. \quad (1)$$

Where $N_{enabled}$ is the number of triggers with the calibration input enabled, $N_{disabled}$ is the number of counts with the calibration input disabled, f is the pulse frequency, and t is the amount of time spent counting triggers. For the tests described here, t = 5s.

Some typical curves from these tests can be seen in figure 4. These curves were produced using $\approx 40mV$, 2.5ns wide, negative gaussian pulses at 10kHz, with the pedestal voltage programmed to 858mV.

The first thing to note is that the offset effect also applies when triggering on pulses. Self-triggering for channel 15 begins when the DAC threshold minus the pedestal is about -20mV, while it begins just below 5mV for channel 17. This is where one begins to see the problem in the offset effect: For a given pulse and pedestal value, each channel has a unique interval of DAC threshold voltages over which it triggers. Unless the pulse is large ($\geq$60mV for 2.5ns pulses when the pedestal is 292mV), some of these intervals will not overlap at all. For the current version of ACDC boards, DAC thresholds and pedestals are set on a chip-by-chip basis. This means it is impossible to to get all channels to trigger on a given pulse unless it is adequately large. More details on this will be given in the minimum amplitude vs. threshold section (Sec 5.1).

Next, note the shape of these curves. The channel goes from not triggering at all to triggering with over 90% efficiency over the course of about a mV or so. After rising, the efficiency tends to stay high until it reaches the noise ceiling. This general shape applies for all of the tests performed.

5.1 Minimum Amplitude vs. DAC Threshold Curves

The idea for the minimum amplitude vs. DAC threshold curves is that for each channel, we keep the pedestal fixed and find the minimum amplitude pulse required to get the desired efficiency as a function of DAC threshold voltage. The program used for these tests is similar to the one made to produce the turn on curves. However, it only recorded the minimum programmed threshold voltages for which trigger rates exceeded the values corresponding to 50%, 75%, 90%, and 98% efficiency. Some typical curves can be seen in figure 5.

In figure 5, note that the curves in each box are roughly linear with a slope of about -2. This slope will vary somewhat with pulse width and pedestal voltage. Next, note that these curves are scattered along the x-axis. This is due to the offset effect. Lastly, note the X's on the end of each curve. For a given channel the X denotes the value of t_{DAC} at which noise rates exceed 200Hz. This is referred to as the "noise ceiling". If t_{DAC} is set higher than the noise ceiling for a particular channel, that channel ought to be masked off to avoid high noise trigger rates.

Curves like these can be used as a tool for determining trigger masks, DAC threshold voltages, and how many channels might be made to work given pulses of a certain amplitude. To use these curves, pick out a DAC threshold-minus-pedestal value and pulse amplitude to get an (x,y) coordinate. The curves located underneath this point will correspond to the channels that will work given those parameters. For instance, the 98% curves show that one could get channels 11 and 12 to work together given a 10mV pulse, channels 9 through 12 with a 40mV pulse, and all 6 with a 60mV pulse.

As mentioned earlier, figure 5 is typical and the curves contained in it were made using pulses that are 2.5ns wide. This pulse width is roughly the pulse width produced by LAPPDs, and as such this figure will give a rough idea for how self-triggering will behave when using the ACDC to readout an LAPPD.

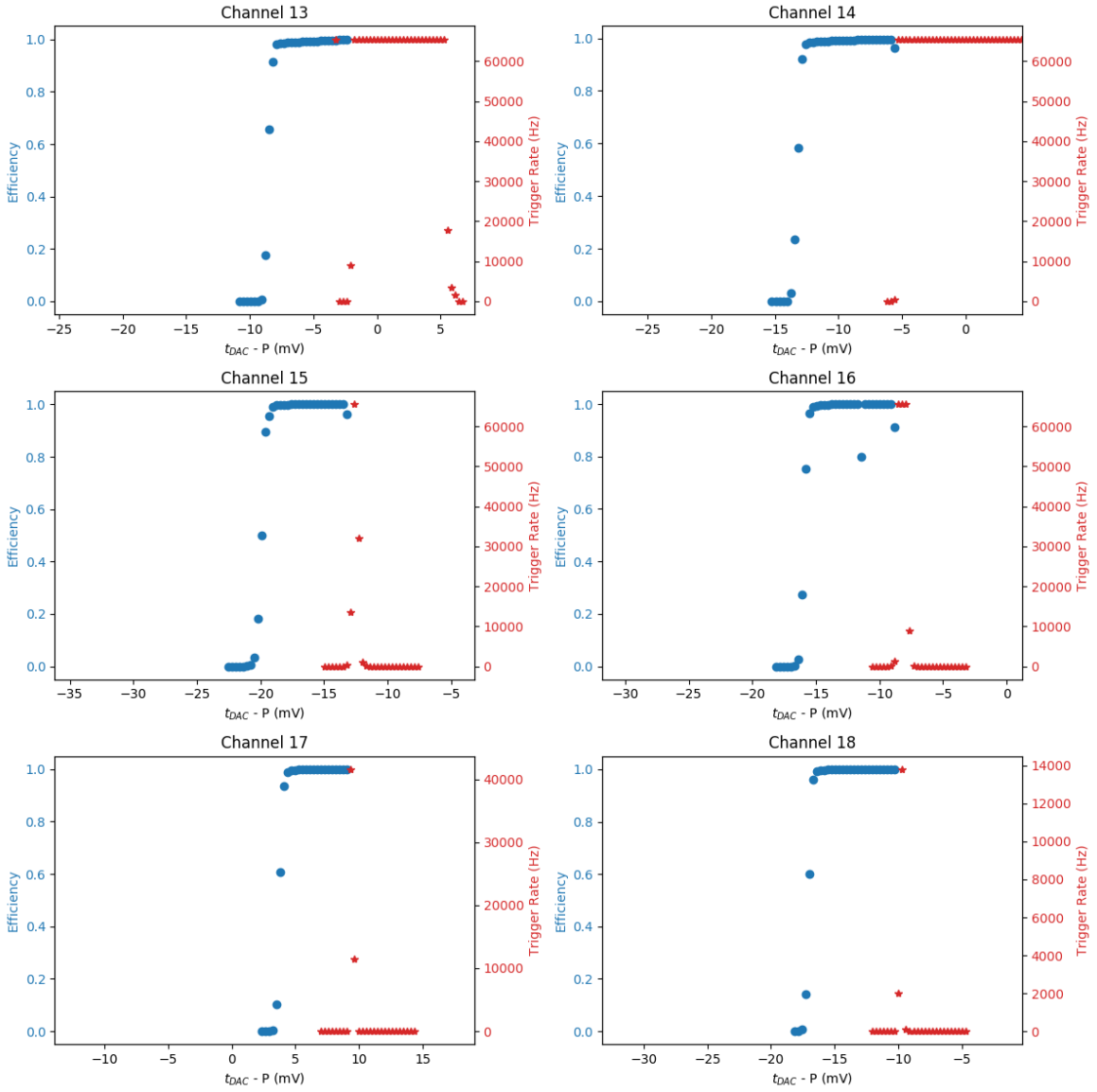

Figure 4: Turn on curves for the channels on chip 3 of ACDC10. In blue is self-triggering efficiency and in red are the noise trigger rates in Hz. These curves were made using $\approx 40mV$, 2.5ns wide, negative Gaussian pulses at 10kHz. This data set was taken with the pedestal set to 858mV.

5.2 Pedestal Dependence

While running other tests, it was found that the pedestal voltage chosen has an effect on self-triggering behavior. To characterize this dependence, measurements were taken using 40mV, 2.5ns, negative Gaussian pulses. For each pedestal value, the program measured the lowest DAC threshold voltage and highest DAC threshold voltage for which the minimum efficiency requirement was met and for which noise rates were below 200Hz. The difference in these two values is called "Trigger Region Width" and is used as a measure for self triggering effectiveness on a channel-by-channel basis.

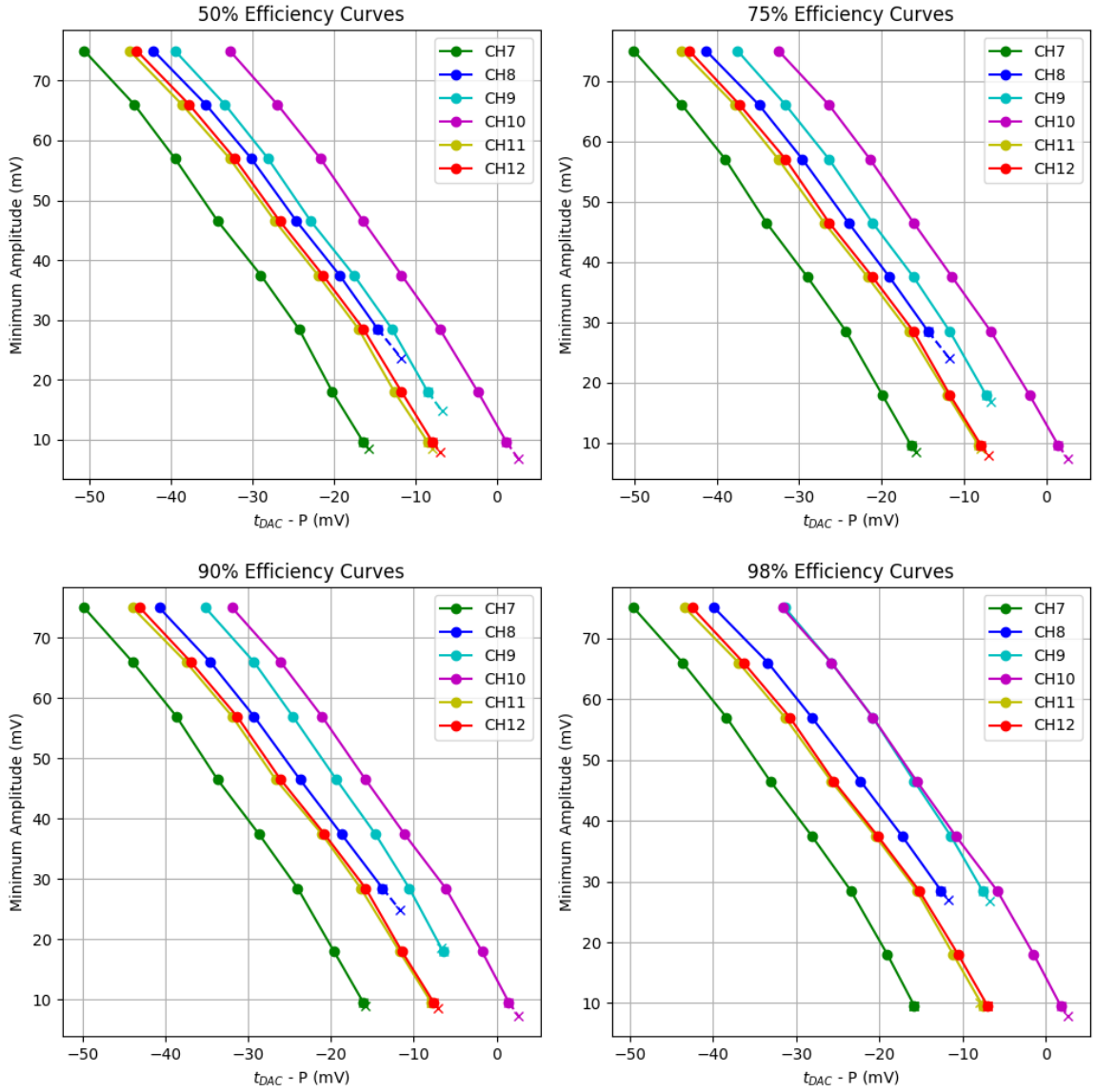

Figure 5: On the x-axis is DAC threshold relative to the pedestal, and on the y-axis is the minimum amplitude pulse required to get the desired self-triggering efficiency at that DAC threshold. These plots are for chip 2 of ACDC20, and made using 2.5ns wide, negative Gaussian pulses at 10kHz, with the pedestal programmed to 292mV.

Figure 6 shows the results of this test. Note that for all channels except 2, the trigger region widths increase as pedestals decrease, reaching their maximum value when the pedestal is 292mV. when the pedestal goes below 292mV, the trigger region width begins to decrease on each of these curves, with the 98% having the biggest drop and 50% having the smallest. While collecting data for the 146mV data points, it was noted that efficiency increased much more slowly with DAC threshold than on data points using a higher pedestal. So, if turn on curves were like those in figure 4 were made at this pedestal, they would have a much more gentle rising edge.

Another feature to note here is that there are no data points for channel 2 on this curve for any pedestals below 439mV. It can be seen here that even though decreasing the pedestal increases trigger region width, it also may prevent the second channel on each of chips from self-triggering on smaller pulses.

It is also worth noting that although using a lower pedestal value increases the trigger region width, it will also reduce headroom when triggering on negative pulses. For instance, a 320mV negative pulse will be cutoff if the pedestal voltage is set to 292mV. However, this trade off may be still be worth it if cutoff pulses can be reconstructed.

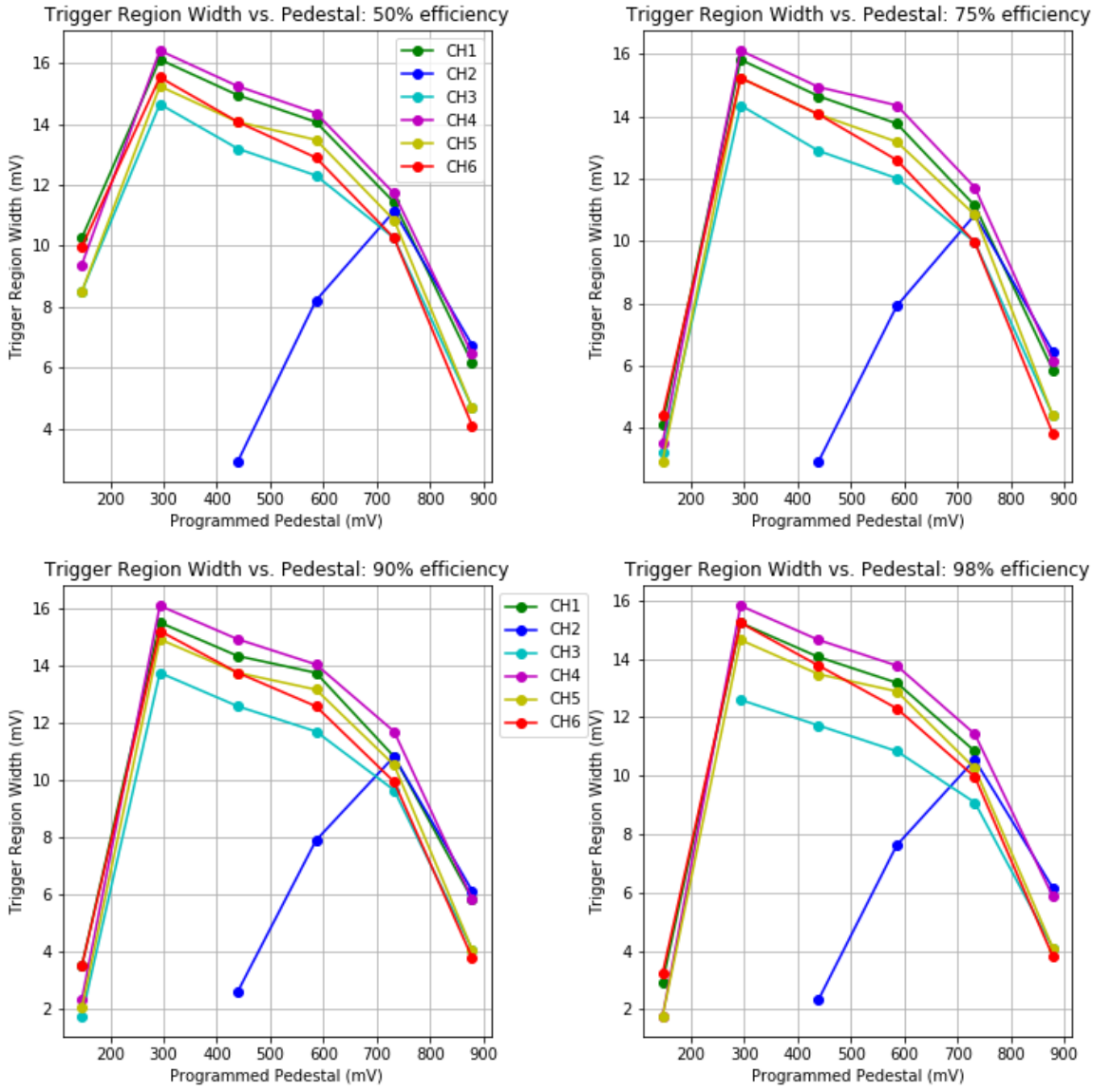

Figure 6: Trigger region width as a function of programmed pedestal voltage in mV. This test was performed using 2.5ns wide, $\approx 40mV$, negative Gaussian pulses at 10kHz.

5.3 Width Dependence

A measurement of triggering sensitivity as a function of input pulse width was also taken. For this test, the pedestal is kept fixed and trigger region width is measured for square pulses of equal amplitude but varying width. The results of this test can be seen in Fig. 7. In this case, it was found that increasing the pulse width from 2.5ns to 25ns increased the self-triggering region width by about 20%. It can also be seen that the curves begin to flatten out as pulse width exceeds 10ns.

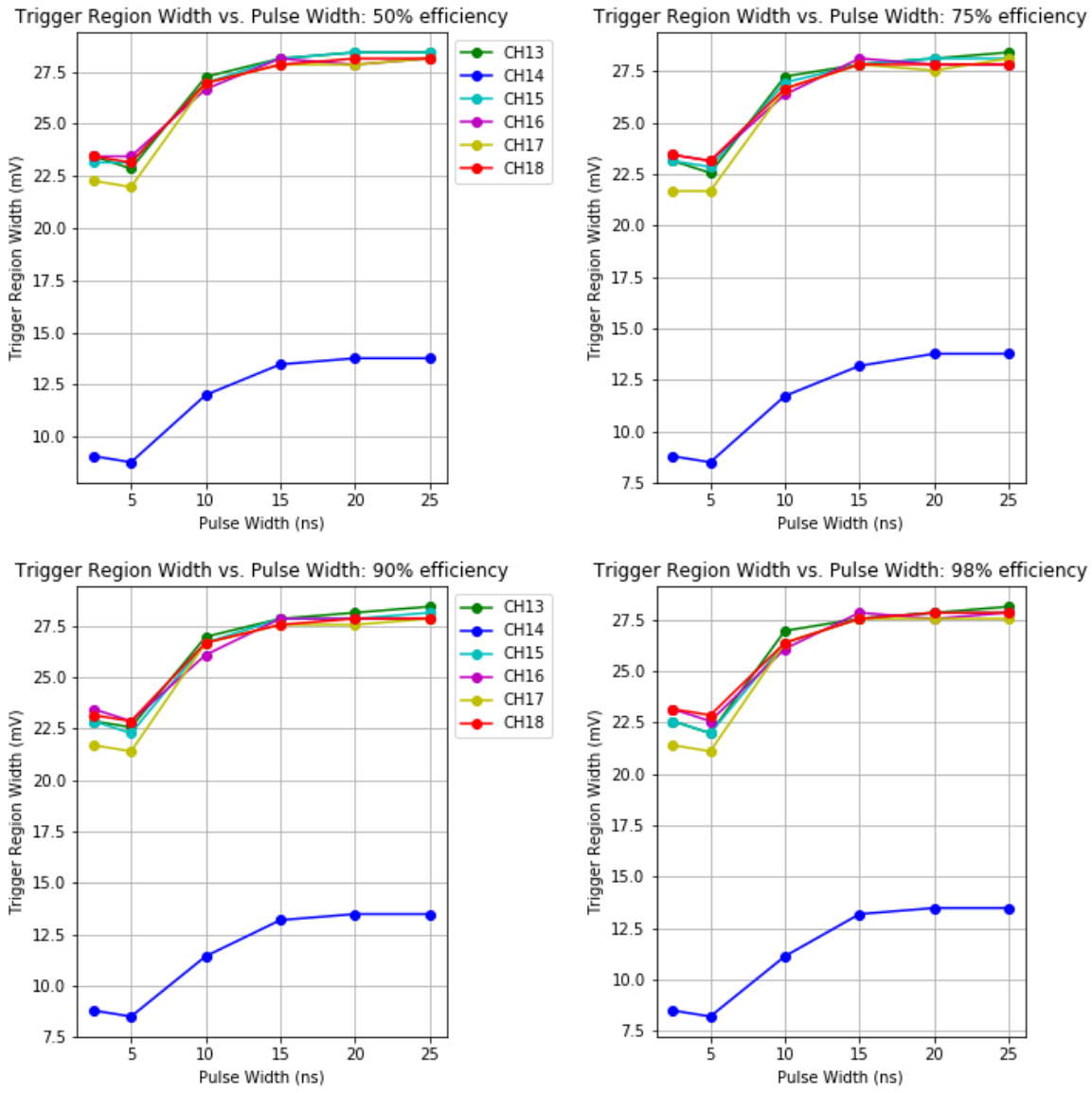

Figure 7: Trigger Region Width (mV) as a function of pulse width. This test was done on chip 3 of ACDC10 with the pedestal set to 292mV. The pulses were $\approx 35mV$ square waves at 10kHz.

5.4 Dependence on Comparator Resistor

Another parameter tested was the value of the resistor connected to the negative pins of the PSEC4 chip's comparators. The idea here is that a smaller comparator resistor will allow more current to flow into the comparator, increasing sensitivity. The standard ACDC board comparator resistor is 1kΩ, which is the value used for all other tests described here. For this test in specific, a 200Ω resistor was used.

To test the dependence on comparator resistor value, minimum amplitude vs. DAC threshold curves were produced with the 200Ω resistor. Figure 8 shows the resulting plots. Aside from the different resistor, these plots were produced using the same parameters as the plots in figure 5. It may not be noticeable by eye, but the slopes in Figure 8 are about -1.6 rather than roughly -1.9 slope when using the 1kΩ comparator resistors. This gentler slope is the advantage gained by using smaller comparator resistors.

Next, note that CH14 is unable to trigger at all on any pulses less than about 45mV. With the 1kΩ resistor, CH14 was able to trigger on pulses around 30mV. So, it can be seen that using a smaller comparator resistor increases the range in which the second channel on each chip triggers on noise.

When using a 1kΩ resistor and requiring 98% triggering efficiency, chip 3 of ACDC20 provided 2 out of 6 channels on a 20mV pulse, 5 out of 6 on a 40mV pulse, and all 6 on a 50mV pulse. With the 200Ω resistor, it provided 2 out of 6 on a 20mV pulse, 4 out of 6 on a 40mV pulse, and required about 65mV to get 6 out of 6 channels. So, using the 200Ω resistor resulted in somewhat worse performance overall in this instance.

6 Conclusions

The most import finding here was the *offset effect*, which causes each channel on any given chip to behave as though it were given a unique pedestal voltage. This happens despite each channel on the chip receiving the same voltages. This may pose a problem when ACDC boards are used, and the user would like to trigger on LAPPD pulses (negative Gaussian, 2.5ns wide) with smaller amplitudes. This is due to the fact that thresholds and pedestals may only be set on a chip-by-chip basis by the ACDC. This problem can be at least partially overcome by using larger pulses, about 40mV or greater. This problem could be completely fixed if a new front end board with more DACs were made. This would allow for either pedestals or DAC thresholds to be set on a channel by channel basis.

a pattern in noise triggering was also found, with the first channel triggering on "noise" over a range of several mV, the second over a range of tens of mV, and the rest over ranges < 2mV. Depending on the pedestal value set, the wide noise triggering on the second channel may prevent it from triggering on smaller pulses.

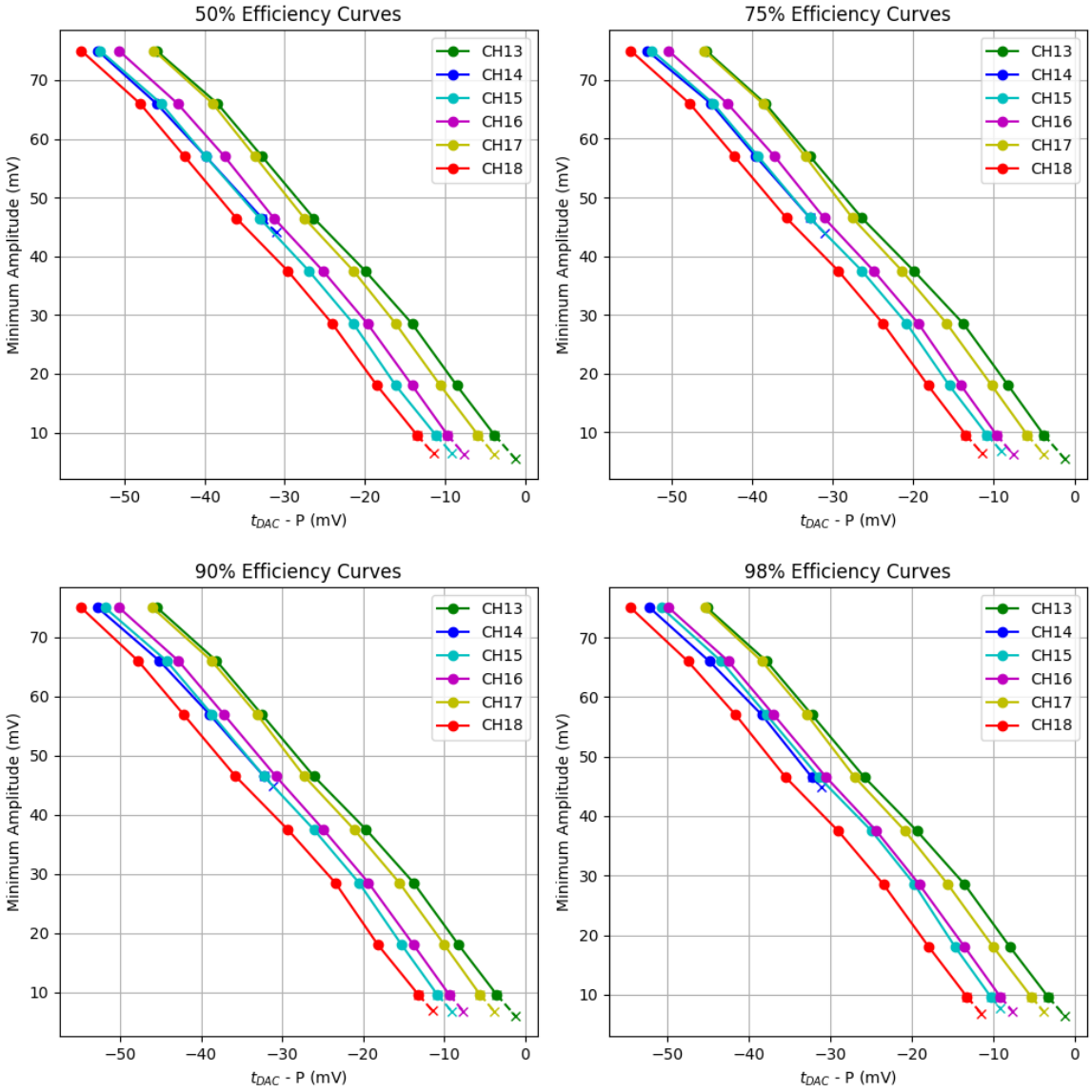

Figure 8: Min Amp vs. t_{DAC} curves for chip 3 of ACDC20. This data was taken with all parameters equal to those of figure 5.1, except with a 200Ω comparator resistor rather than 1kΩ.

Pedestal voltage was found to have an effect on self-triggering, with the largest trigger region widths occurring when the pedestal is set to 292mV. Using a smaller pedestal voltage will reduce headroom, with larger pulses being cutoff. However, this drawback may be overcome by using pulse reconstruction techniques.

Self-triggering effectiveness was found to have a dependence on pulse width. During testing, it was found that increasing the width of a square pulse from 2.5ns to 25ns increased the trigger region width by about 20%. It was also found that the trigger region width vs. pulse width curves tend to flatten out when pulse width exceeds 10ns.

Lastly, self-triggering behavior was found to depend on the value of comparator resistor used. Using a 200Ω resistor rather than 1kΩ was found to decrease minimum amplitude vs. DAC threshold slope absolute values from 1.9 to 1.6, at the expense of decreased performance on the second channel of the chip. In the test performed, this resulted in slightly worse performance overall.

7 Acknowledgments

Thanks to Evan Angelico, Mircea Bogdan, Jonathan Eisch, Henry Frisch, John Judge, Eric Oberla, Mark Zaskowski.

Appendix A: Firmware

A.1 ACDC firmware

The ACDC firmware used for these tests can be found at `https://github.com/lappd-daq/ACDC-firmware-25MHz` . The version of the firmware linked to here includes adjustments made for "scalar mode", which allows for self triggers to be counted. This particular version is also meant to be used with 25MHz clocks coming from the ACC, and a 40MHz oscillator on the ACDC.

A.2 ACC firmware

The ACC firmware used in these tests can be found at `https://github.com/lappd-daq/ACC-2xSPF` .

Appendix B: Software

The software we used can be found at `https://github.com/lappd-daq/acdc-daq/tree/acc-acdcdaq`

B.1 Changes to LogData.cpp

LogData.cpp was edited to add on a feature for scalar mode. To use LogData for taking scalar measurements, go into your trigger config file and set rate only mode to 1. After that, go through your configuration as normal. Once everything is configured, run LogData with 2 as the final argument. This will send a software trigger to the ACDC board every 3 seconds or so. Once data has been collected, rates will be written to the filename you entered with .acdc.rate tacked on to the end of it. Data will be written to your file in columns. The first column is event number, the second is board number, the third is a time stamp, and the others are trigger rates for each channel. The trigger rates part tells you the number of self-triggers counted in 1 second.

B.2 Automation.cpp

Automation.cpp measures the noise turn-on curve for each channel on an ACDC board, or a specific subset of them if desired. Pedestal voltage (in DAC counts), board number, save file name, and channel mask are taken as user input arguments. The channel mask is optional, and will include all channels on the board by default. An example input can be seen below.

```
./bin/Automation 1000 3 Noise_Test 20820820
```

The first three arguments say that we'll test board 3, set the pedestal to 1000 DAC counts, and call our file Noise_Test.dat. The last argument is a hex word that tells the program only to test channels 6, 12, 18, 24, and 30.

After being told to run, Automation.cpp sets the trigger mask only to look at the first channel in its list and proceeds to scan through thresholds from 250 DAC counts below pedestal to 250 DAC counts above pedestal, incrementing in steps of 5 DAC counts. At each point, it measures the number of self triggers counted in a one second interval. When the self-triggers reach a significant, non-zero value, the script begins a fine-grained scan starting from a few counts before the significant data point. After the fine-grained scan is complete, the code moves on to the next channel on the list.

This program writes data to the file in three columns. The first column is channel number, the second column is threshold (referred to as t_{DAC} in the bulk of the paper) in DAC counts, and the third column is the number of self-triggers counted in one second at the given threshold.

B.2.1 Warnings

In the event of a major communication error part way through data collection, the boards and computer should be power cycled, and the process started from the beginning. If a major communication error occurs immediately after starting the program, just run the program again. Make sure to restart the boards and computer between taking data sets. I found that running the program more than once without restarting can cause bugs. This isn't ideal, but I'm unsure how to fix it.

Lastly, make sure to allow RO frequencies to settle in before taking any data sets. I found that running while the frequencies are still adjusting causes extra noise triggers.

B.3 Automated_SelfTrig.cpp

Automated_SelfTrig.cpp is meant to measure turn on curves for pulse triggering. To use this program, hook up a function generator to the calibration input, which is the SMA jack labeled J4 on the ACDC board. The program takes Pedestal (in DAC counts), board number, pulse amplitude (mV), savefile, and channel mask as inputs. If no channel mask is given, the code tests all channels on the board. Here is an example input:

```
./bin/Automated_SelfTrig 1000 3 10 Test_Triggering 20820820
```

The first four arguments say that want our pedestal to be pedestal is 1000 DAC counts, we want to test board 3, our pulse will be 10mV, and that we want to name our file Test_Triggering.dat. The final input gives the channel mask, which says to test only channels 6, 12, 18, 24, and 30.

After being told to start, Automated_SelfTrig begins scanning through thresholds starting from pedestal - 2*amplitude (DAC counts) or pedestal-100 (DAC counts), with the smaller value being the one used. From there, it proceeds to scan in a way similar to Automation.cpp, except with the calibration input enabled. When the program sees a significant, non-zero value it begins its fine-grained scan. In the fine-grained scan, the code disables the calibration input and counts the number of self-triggers over 5 one second intervals, giving the noise trigger rate. It then enables the calibration input and does the same measurement, giving the rate of pulse-triggers plus noise triggers. Once the program sees the noise rate exceed 200Hz, it stops and proceeds to the next channel.

Data is written to the file in three columns. The first column is threshold voltage in DAC counts. The second column is the number of noise-trigger counts recorded over 5 one second intervals. The third column is the number of triggers counted over 5 one second intervals with the calibration input enabled. this data can be used to estimate self-triggering at each point using the expression in equation 1.

B.3.1 Warnings

The same warnings as Automation.cpp apply. Let the RO frequency settle before running the code, and make sure to restart your boards and computer in between data sets.

The calibration input on the ACDC board will attenuate pulses significantly. For the boards I used, I found this to be a factor of about 15 or so. I recommend calibrating some channels on your board, then using those to measure the pulse amplitudes post attenuation.

In addition, it is also the case that noise triggering may differ depending on whether the calibration input is enabled or disabled. I recommend running Automated_SelfTrig without putting in any signals as a way of comparing the two.

B.4 Automated_SelfTrig_Cutoffs.cpp

Automated_SelfTrig_Cutoffs works similarly to Automated_SelfTrig, except it only records the data points for which trigger rates exceed values meant to correspond to 50%, 75%, 90%, and 98% efficiency. For instance, when we used 10kHz pulses, it would record the minimum thresholds for which the number of counts with the calibration input enabled exceeded 25000, 37500, 45000, and 49000. This program takes pedestal (DAC counts), board number, pulse ampltiude (mV), pulse frequency (Hz), savefile name, and channel mask (optional hex word) as inputs. Here's an example input:

```
./bin/Automated_SelfTrig_Cutoffs 1000 3 10 10000 T 20820820
```

The first five arguments give the pedestal (DAC counts), board number, pulse amplitude (mV), frequency (Hz), and filename respectively. The last gives your desired channel mask.

Typically, the most interesting part of the turn on curves generated by Automated_SelfTrig are the thresholds at which self-triggering "turns on". This code doesn't take nearly as long as Automated_SelfTrig, and allows you measure those thresholds and corresponding trigger rates. In addition to running with pulses, we also recommend running this test without pulses to measure the thresholds at which noise trigger begins on each channel.

Data is written to the file in three columns. The first column is threshold (in DAC counts), the second is the number of noise triggers counted over 5 one second intervals, and the third is the number of triggers with the calibration input enabled over 5 one second intervals. Moreover, data is reported in chunks of four rows. The first row corresponds to 50% efficiency, the second to 75%, and so on.

B.4.1 Warnings

The same warnings that apply to Automated_SelfTrig also apply to this program.

Appendix C: Hardware

Advice on physically setting up a DAQ like the one used here can be found in the appendix of the following paper: `http://lappddocs.uchicago.edu/documents/316/sendit`

References

[1] Frisch, Henry (2016). The LAPPD Pico-second Photo-Detector PowerPoint presentation at MIT Lincoln

[2] `http://ieeexplore.ieee.org/document/12695/`

[3] G. Haller, B. Wooley, A 700 MHz Switched Capacitor Analog Waveform Sampling Circuit, SLAC-PUB-6414, 1993. `http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.433.9386&rep=rep1&type=pdf` Labs. `http://www.bostonphotonics.org/files/seminars/OSW2016_HFrisch.pdf`

[4] Oberla et al. (2013). A 15 GSa/s, 1.5 GHz bandwidth waveform digitizing ASIC `http://lappddocs.uchicago.edu/documents/218`

[5] Bogdan et al. (2016). A Modular Data Acquisition System using the 10 GSa/s PSEC4 Waveform Recording Chip `http://lappddocs.uchicago.edu/documents/288`

[6] Bogdan et al. (2014). A Data Acquisition System using the 10 GSa/s PSEC4 Waveform Digitizing ASIC http://lappddocs.uchicago.edu/documents/265/sendit

[7] http://edg.uchicago.edu/~bogdan/AnniesCentralCard/schematics.html

[8] https://hep.uchicago.edu/~eric/?/pcb/acdcrevB/