Improved Approach for Utilization of FPGA Technology into DAQ, DSP, and Computing Applications

CONTENTS

From Phase I Application to list objectives

Part a. Identification and Significance of the Problem or Opportunity, and Technical Approach

IDENTIFICATION AND SIGNIFICANCE OF PROBLEM

Advanced data acquisition (DAQ) systems, signal processing, and data preprocessing technologies are critical to particle detectors and associated systems. Two extremes exist in high energy physics. One is obvious, which is the requirement for very large systems to handle multi-gigabyte/second data rates. However, to develop these systems, a much smaller flexible prototype system is required to develop the final product. A hardware and software environment that allows quick development and minimal change is needed to move to the final product, which will have many copies produced. Innovation Partners believes an improved approach is possible and will be described in this proposal. In particular, current FPGA technology can allow for preprocessing Analog-to-Digital Converters (ADCs) in real time. This includes ADCs that do conversions in 25–50 ns, meaning that signals can be digitized at an early level and then the FPGA can work with the digital information with no further analog stages.

The opportunity to produce a standard hardware/software solution to manufacturing DAQ components such as trigger modules would greatly decrease the expense and time to create this part of the DAQ system. There are aspects of new FPGAs that improve their usefulness in providing reconfigurable logic for a trigger. One is their ability to be reprogrammed in real time. This would mean that one could change the trigger logic rapidly based on changing conditions. New FPGAs can be at least partially reprogrammed in milliseconds or even a single clock cycle. This is something that no NIM, CAMAC, or custom logic boards can provide. Inclusion of FPGA solutions should often replace the need for custom logic boards. Another ability of FPGAs is that they have flexible clock locking. For example, QuickLogic®manufactures an FPGA that can have multiple clocks, from 5–20, in multiple areas of the FPGA. These can also be independently be programmed to multiply or divide the clock speed, or change the phase of the clock. This provides the DAQ designer capabilities not possible in earlier technology. These FPGAs also offer "instant on" capabilities that older SRAM based designs do not have. A side benefit is that these devices are free from what is termed "Single Event Upset" where the reprogramming process can generate errors in the code that actuall gets loaded. This same company makes FPGAs that have 10 microamp standby currents and only consume 60 mW at 175 MHz.

Data throughput can also be very high. They can be uses as First-In-First-Out (FIFO) buffers at several hundred MHz. With 250 to over 650 inputs, FPGAs offer other powerful logic capabilities. Also, large capacity FPGAs are available with more than 1,000,000 gates.

OPPORTUNITY

Innovation Partners recognizes that recent advances in new digital signal processing techniques, in particular FPGAs, provide significant improvements in high-performance DAQ, computing, and signal processing systems. What is lacking is software/hardware co-designs providing forward

compatibility so that new FPGA hardware can be adopted with minimal impact. Some systems are being developed by various companies such as National Instruments, but the solutions are for propriortory devices and often prohibitively expensive. The company sees this as an opportunity to help increase the capabilities and quality of DAQ and signal processing capability. Research under this proposal will study improvements possible in DAQ algorithm development, DAQ upgrades/migration, and signal conditioning or processing of data before sending them to the next level. For example, event trigger generators could make use of FPGA results. Since the FPGAs are reprogrammable, this solution allows changes to be made for different detector conditions or to implement other necessary changes. Taking advantage of this new hardware and providing an environment where the programmer is able to use a seamless system provides the opportunity to:

- decrease significantly the time to exploit new FPGA designs.
- keep already developed software, avoiding duplication of effort.
- provide at least a 100% performance improvement.
- provide at least a 100% improvement in life-cycle costs.
- provide the same functionality of an application-specific integrated circuit (ASIC) design, but at significantly lower costs.
- use a commercial off-the-shelf (COTS) design to improve manufacturability.
- take advantage of work for things like software radio to find high performance COTS designs.
- reduce the cost of updating new hardware.
- provide excellent capabilities for scaling to the needs of larger detector systems and upgrades.
- provide FPGA computing for data manipulation with superior human-hardware interface.
- provide a straightforward path to adding new digitization technology.
- allow the addition of non-FPGA technology for pre- or post-processing signals.

TECHNICAL APPROACH

Innovation Partners proposes a software/hardware co-design approach to reduce both the difficulty and time implemention of FPGA solutions to data acquisition and specialized computational applications. At present, FPGAs can require excessive time for programming and require specialized knowledge; the company's solution will greatly reduce both. Not only are FPGAs ideal for DAQ and embedded solutions, they can also be the best solution for specialized signal processing to complement or replace digital signal processors (DSPs). By allowing FPGA programming to be done via a GUI interface, with the equivalent of a simple compilation, algorithm changes and improvements can be easily implemented decreasing the life-cycle costs and allowing substitution of new FPGA designs to maximize hardware independence. The basic process-overview of the company's approach is shown in Figure 1.

Part b. Anticipated Public Benefits

TECHNOLOGICAL AND ECONOMIC BENEFITS

The proposed technology system will save development time, improve performance and reduce the cost of programming and updating FPGA devices. At present, FPGAs are used extensively in data acquisition, signal processing, and embedded systems. These are critical components in systems used by experimental physics programs. Innovation Partners will provide affordable systems where scientists can utilize new FPGAs in a mode almost transparent to the algorithm developer.

3 Degree to which Phase I has Demonstrated Technical Feasibility

The results for Phase I research were very exciting. The proposed methodology was shown to work very well. Several different common devices were created in an FPGA using the proposed software/hardware model proposed in Phase I. One example was a Constant Fraction Discriminator. This type of discriminator is designed to improve timing measurements by triggering on a constant fraction of the height of the electrical pulse. A discriminator that triggers at a specific voltage threshold will vary in its timing if the pulse height is different from pulse-to-pulse. This should be obvious when comparing a large to a small pulse. The large pulse will pass the threshold early on the leading edge, whereas a small pulse may only pass the threshold near the maximum of the pulse. This can result in variations in the timing of the pulses on the order of the total rise time of the pulse. By triggering on a constant fraction of the maximum of the pulse, this problem is avoided, giving a much better timing resolution.

The FPGA CFD was able to be programmed in less than one day and showed an improvement of at least a factor of five over the leading edge version. One aspect of such an implementation is that CFDs tend to be expensive and bulky. The FPGA version is inherently low cost and can be added to any TDC configuration with a minimal amount of work.

Several other common functions used in data acquisition systems were created. All performed better than expected and most took less than a day to program. Thus it was shown in Phase I that the software/hardware approach could permit an enormous amount of flexibility. Furthermore, changes to the pulse shaping software or addition of new methods were shown to be simple to implement by undergraduate students with little to no experience with FPGAs.

3.1 Feasibility questions

- **FPGA Design Automation Focus:**
  - Is it possible to reduce the programming time from 6–18 months down to days?
  - Is it possible to enable a non hardware engineer to program the FPGAs in days?
  - Is the quality of the FPGA design high enough for production use? (speed and area)
- **Data Acquisition Focus:** Can Concurrent EDA's tools produce a hardware design that handles data at 100 million samples/second without loss of data?
- **Results Processing Focus:** Given a set of results from a significant number of data acquisition channels, can the processing of this set of results be performed in an FPGA in real-time AND programmed by a non hardware engineer? Specifically, can Principal Component Analysis be implemented in software and automatically converted to an FPGA design using Concurrent EDA's tools?

3.1.1 The Need for FPGA Processing in Data Acquisition Systems

Data acquisition hardware heavily utilizes the power of FPGAs to enable real-time processing. Figure 1 is a DAQ card from National Instruments that samples two channels at 100 MS/s (million samples per second) with 14-bit resolution. For this card, as with most DAQ cards, the FPGA is directly connected to the A/D chip. Data is fed into the FPGA at the same clock rate as the sample rate. The FPGA logic can then operate on this stream of data and as long as the logic is pipelined and does not stall, the FPGA can process the A/D at peak rates. The output of the FPGA logic is typically a FIFO that is then fed into on-board memory. This memory is then accessed across the bus interface by the host processor. In a National Instrument's system, the host processor can be programmed in ANSI C using the NI CVI software or it can be programmed in the LabVIEW graphical programming environment. This system enables the DAQ card to be fed into a large system and in this way, custom measurement system can be created.

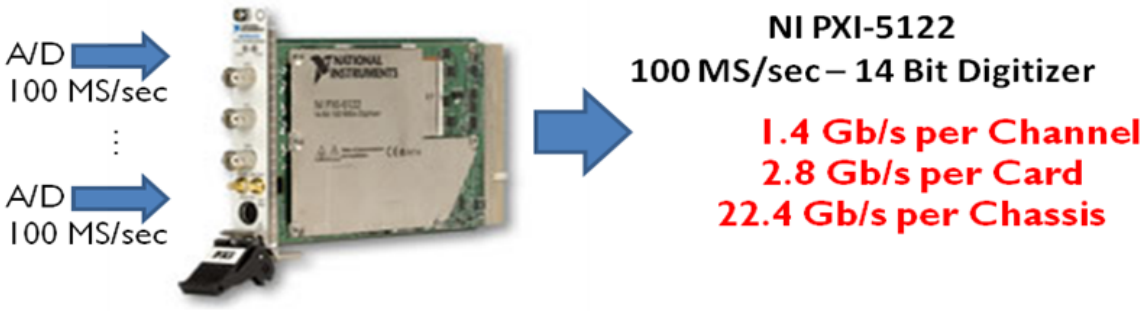

Figure 1: National Instruments Data Acquisition Card operating at 100 million samples per second on two channels. This generates 2.8 Gb/s of data per card and 47.6 Gb/s per chassis. An on-board FPGA enables processing at these peak rates without losing data.

Data acquisition systems require real-time processing. Given that the data is being read in at 100MS/s over two channels, the processing capability must be able to handle a sustained 2.4 Gb/s. If a process is to be used, and 10 instructions, for example, are required for each bit of data, then the processor would have to operate at 10×100 MS/s or 1 GHz. An FPGA, however, can be pipelined so that all 10 "instructions" are executed in parallel on different data samples and thus, only a 100MHz clock is required. Additionally, FPGA pipelined computations scale well and can execute thousands of instructions in parallel. Processors, on the other hand, do not scale well as they only execute a few instruction per clock cycle. Even if a 2 GHz processor was able to execute 5 instructions per cycle, then a maximum of 100 instructions would be allowed per data sample. A cache miss or an interruption by the operating system would cause data to be lost as both of these events can require more than 50 ns. Additionally, these calculations are on a per-channel basis and thus, a 1GHz processor would be needed per channel. For an FPGA, each channel can have its own processing pipeline and thus, the FPGA scales extremely well.

Processing on the FPGA is required because DAQ systems have to handle large data rates that often exceed traditional bus bandwidths. For example, Figure 1 shows how a single card can generate 2.4 Gb/s of data using just two channels. Even a small 10 channel

system generates 14Gb/s of data traffic. This is more than a traditional PCI bus can handle. Since Compact PCI and PXI are all based on PCI, this is problematic. For high-end bus technology, such as 32-lane PCI Express (PCIe), the data capacity is expanded to 64 Gb/s. Even this bandwidth is exhausted when 64 channels are utilized as this would require nearly 90Gb/s of *sustained* throughput. It is clearly imperative that processing be performed by the FPGA on the DAQ card itself.

3.2 The FPGA Design Problem

The problem, however, is that the traditional FPGA design flow is not well suited to scientists. Scientists are gifted individuals who have great depth in their domain of expertise and utilize custom software in their experiments. Unfortunately, the traditional FPGA design flow does not capitalize on this capability and requires that a *hardware description language* (e.g. VHDL and Verilog) be used to design the FPGA. VHDL and Verilog are not just another syntax for creating sequential programs but are low-level circuit description languages that require the designer to have a detailed understanding of parallel logic design. Additionally, these languages are technology dependent and have to be rewritten for each new FPGA device. For example, the code for a 100MHz FPGA does not work well for a newer 200 MHz FPGA as the technology changes and new hardware resources, such as multiply-accumulate blocks, are available to the designer. To achieve peak FPGA processing rates, as is needed by DAQ systems, the designers must exploit all of the available resources in the FPGA.

FPGA designs are complex and are labor intensive. In 2005, EE Times performed a study of the design duration and costs associated with creating FPGA designs. As shown in Figure 2, only 9% of the designs were completed in 3 months, 17% in 3 to 6 months, and 47% required 6 to 12 months. 27% required more than a year to complete. The median design time was 9 months.

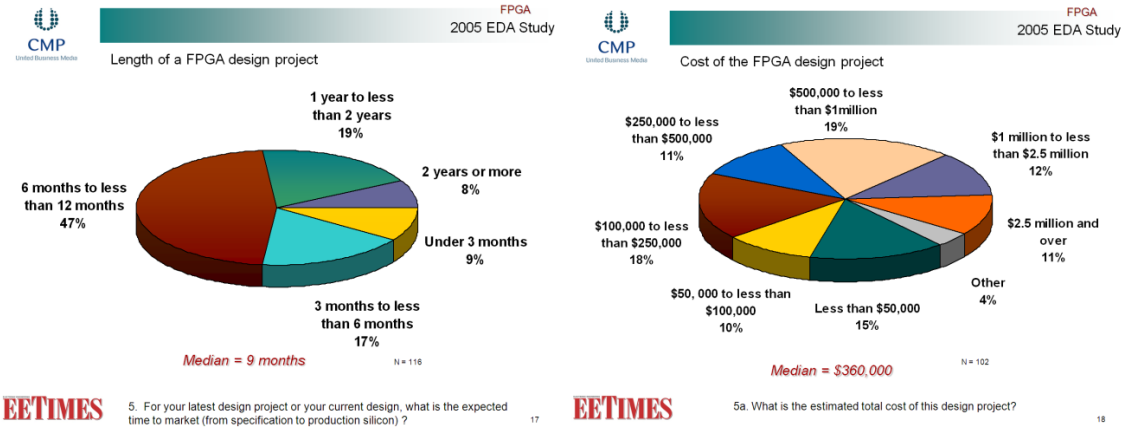

Figure 2: FPGA designs are both lengthy (9 months median length) and expensive ($360K median cost), according to the 2005, EE Times FPGA Survey.

FPGA designs are also expensive. According to the same EE Times survey, the median cost of an FPGA design was $360K. Almost half (48%) of the FPGA designs cost between

$100K and $1M. 25% cost less than $100K and 23% cost over $1M.

With 90,000 FPGA designs started each year, there is a significant market opportunity if the design time and design costs can be significantly reduced.

3.2.1 Why are FPGAs so difficult to design?

At the core of the problem is the level at which designers are forced to work. An FPGA designer had to determine how much logic can be placed between every pair of registers such that the delay through this logic does not slow down the entire circuit. If a design incorrectly designs a single set of logic, the *entire* design must be slowed down. Since FPGAs have a single clock, a single error in the design causes the entire FPGA to perform slowly. This process is tedious and error prone.

Verification is also a lengthy portion of the FPGA design process. Typically, the domain expert writes software to implement their algorithms. The design engineer then learns the algorithm and *manually* converts the functionality of the software into parallel hardware. This requires that the design engineer have a detailed understanding of the algorithm. Verifying that the parallel hardware performs the exact same computation as the software is difficult as they were written in two entirely different languages with different design tools/compilers.

3.2.2 The Solution: The DAQ Accelerator Design Tool

As part of Phase I of this project, Concurrent EDA has customized their tools to dramatically improve the speed of FPGA design for DAQ signal processing. This new tool is called the **DAQ Accelerator** as it enables software to be transformed into hardware computations in an FPGA-based data acquisition card. Rather than requiring hardware engineers to design at the logic-level, the DAQ Accelerator enables physics domain experts to program the DAQ card using standard C software. The DAQ Accelerator enables the physicist to view the FPGA DAQ hardware as a software function that is executed on every A/D sample. A history of prior samples is also provided to the user so that they can view up to 64 points on the analog waveform per sample.

Figure 3 shows the hardware framework for the DAQ Accelerator. In this diagram, the A/D chip is directly connected to the FPGA with its output labeled as A/D Samples. A series of registers within the FPGA buffer samples of the A/D to enable the user to access prior samples. The user sees all A/D samples as an array called DAC[], where DAC[t] is the A/D sample from t cycles ago. Free running counters are also available to enable the user to change the sampling rate or for timing calculations. The output of the logic is two signals, Valid and Result. If Valid is 1, then Result is placed into the FIFO, otherwise the current sample (DAC[0]) is not kept. The data from the FIFO is sent to the host processor for further processing. In Phase I, we demonstrated that this process can be fully integrated into the National Instruments DAQ FPGA hardware and integrated into the LabVIEW host software.

The code in Figure 4 shows how samples that are close to zero are detected and ignored. Figure 5 is slightly more complex as it also incorporates a value from the knob on the LabVIEW console. This example demonstrates that configuration data from LabVIEW can be sent into the FPGA to configure the user-specified computation. From a programming

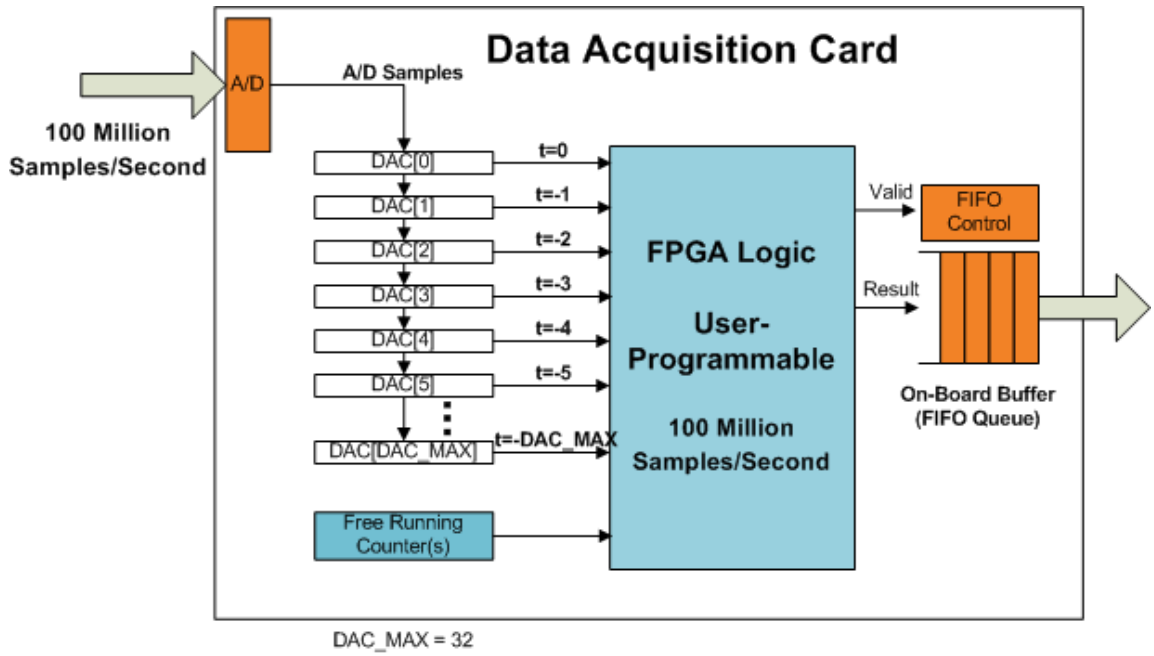

Figure 3: Concurrent EDA's DAQ Accelerator Hardware Framework. This diagram shows how samples from the A/D are fed into the User Programmable FPGA Logic. The out of the User Logic is fed into a FIFO that interfaces with the DAQ Hardware.

perspective, these values are seen as a set of 32 global variables called `WRITE_REG0` through `WRITE_REG31`. Addition `READ_REG` values are also available to the user and are then available within LabVIEW as a scalar value.

Concurrent EDA's DAQ Accelerator is focused on enabling non-computer engineers to create high performance DAQ hardware. This tool is focused on scientists, who have the best knowledge of the experiments but don't want to design circuits just to conduct an experiment. However, the quality of the results must not suffer and the automatically generated circuits must be able to keep up with the A/D data rates. The user interface is the same as that of a software compiler except that the final step requires an FPGA place-and-route run, which can take hours. To enable this last lengthy step to be performed only once, DAQ Accelerator provides early area estimates for the user. As such, the user can explore multiple algorithms quickly and selectively determine which ones need to be downloaded onto the FPGA. See the DAQ Accelerator workflow detailed in Figure 6

3.2.3 How is this possible?

Compiled software is a sequence of simple processor instructions that perform operations on data stored in a set of software registers. If one instruction performs a computation and places the results in register R1 and another instruction reads R1, then there is a data dependency between the instructions. By creating a graph of these low-level instructions and data dependencies, the computation can be *unrolled* and turned into hardware. Control flow (e.g. if-then statements) can be turned into data flow through a process called *predicted*

```
1  #include "DAC.h"
2
3  /* Average of all samples in DAC sample space */
4  DAC_FUNC {
5
6      if (DAC[0] < (float)0.001 && DAC[0] > (float)-0.001) {
7          VALID = 0;
8      } else {
9          DAC_RESULT[0] = DAC[0];
10         VALID = 1;
11     }
12 }
```

Figure 4: Concurrent EDA's DAQ Accelerator enables signal processing software to be converted into FPGA hardware designs in hours rather than months.

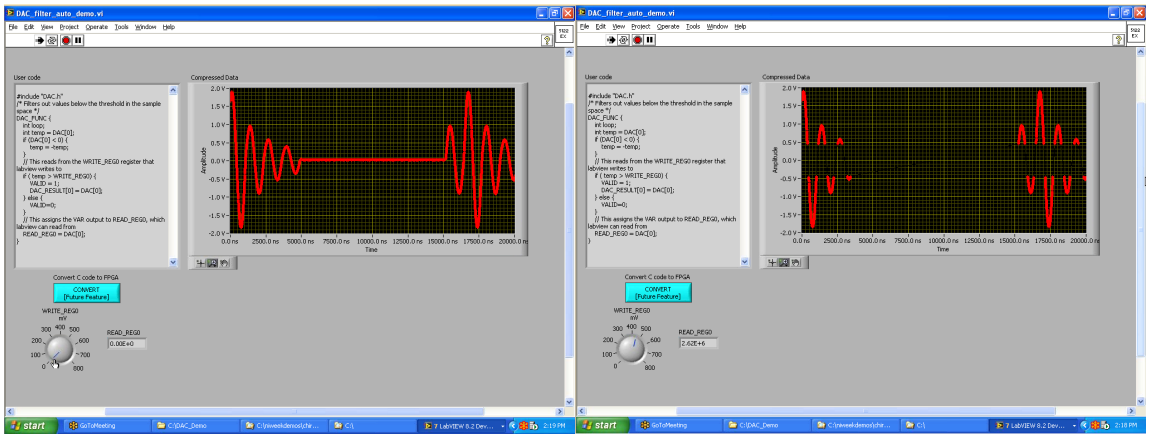

Figure 5: Demonstration of a Zero Suppression algorithm running on the FPGA in a National Instrument's DAQ card and displayed in LabVIEW on the host processor. In this example, the knob changes the height of the "zero" band.

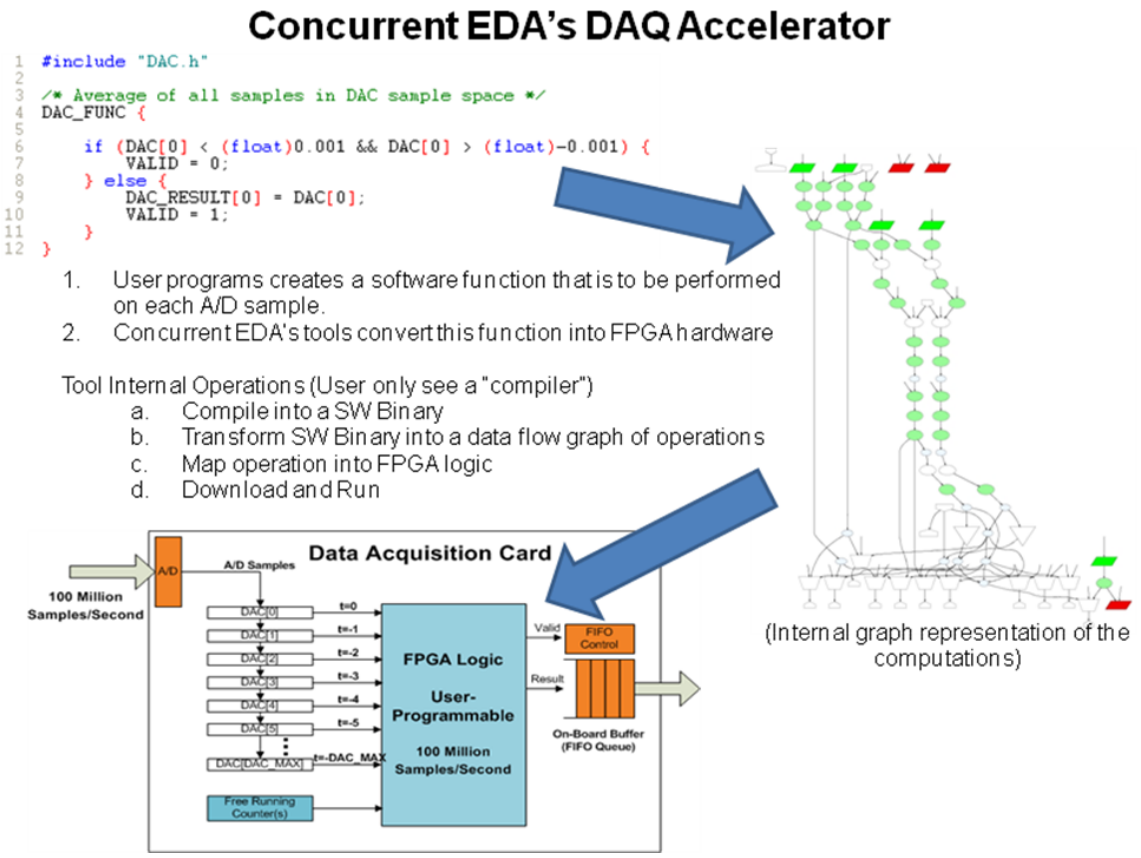

Figure 6: Concurrent EDA's DAQ Accelerator transforms user-defined software functions into a low-level data flow graph of operations. The graph above is the data flow graph of the zero-suppression program shown earlier. Using static timing analysis, this graph is converted into FPGA logic and converted into an FPGA bit-stream.

execution. In this way, an entire function can be converted into a data flow graph. Since we are targeting DAQ hardware, the inputs and outputs are already defined. The graph is then converted into a hardware data flow graph and operation-level timing information is incorporated into the graph. Static timing analysis is performed on this graph and registers are inserted to achieve the user-specified A/D sampling rate. The graph is then used to create VHDL that is then transformed into an FPGA bit-stream using the Xilinx (or Altera) backend tools.

3.2.4 What are the limitations of this approach?

There are no programming language restrictions within the *DAQ_func()* function. However, FPGA area is not unlimited and can be exhausted. Loops are allowed and are unrolled to create a pipelined hardware block. Large loops will require a large amount of FPGA hardware and may exceed the area available in the FPGA. Complex control flow commands, such as nested if-then-else structures and case/switch structures are allowed and can be as complex as needed. Both integer and floating point computations are allowed but require different amounts of FPGA area.

There are some function-level features that are available in software that do not work in version 1.0 of the DAQ Accelerator tool. Recursive calls from *DAQ_func()* to itself are not allowed. However, fixed depth recursion can be converted in theory and may be included as a feature later. Since this is not required and since the user has access to a large array of prior A/D samples to implement recursive-like calls, this is currently a low priority feature. Function calls from within *DAQ_func()* to other functions is currently supported using the #inline command in the software. Future version will enable this automatically. Support for precompiled libraries is a feature that will be implemented in Phase II.

3.3 Timing-related examples

For various timing benchmarks, Landau shaped pulses were used. See Figure 7 for an example Landau shaped pulse.

3.3.1 Leading edge discriminator

One of the simplest methods for extracting timing information from a detector pulse is a threshold discriminator followed by a TDC. This example approximates this functionality of the threshhold discriminator using a high-speed ADC and FPGA. The basic premise is to analyze streaming input samples from the ADC and generate a timing signal when the sampled waveform value exceeds a set threshold.

3.3.2 Constant fraction discriminator

A more interesting tool for extracting timing information is a constant fraction discriminator. A CFD offers significanly better timing resolution than a leading edge discriminator because it addresses the issue of time-walk. Walk in this sense is a systematic shift in the timing pulse related to the amplitude of the input pulse. A CFD addresses this problem by triggering when a pulse is a certain fraction of its maximum amplitude.

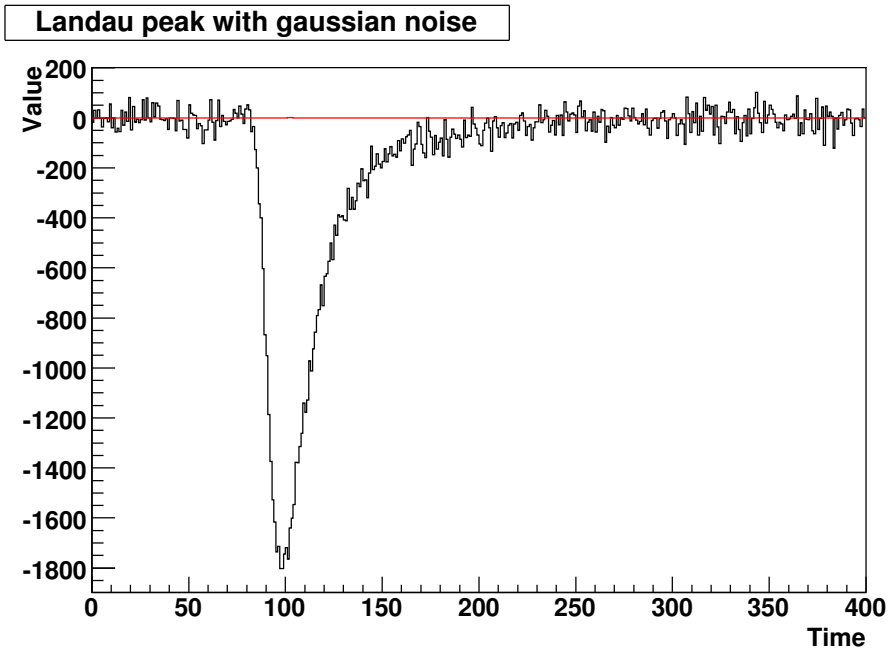

Figure 7: Simulated Landau pulse with Gaussian noise added.

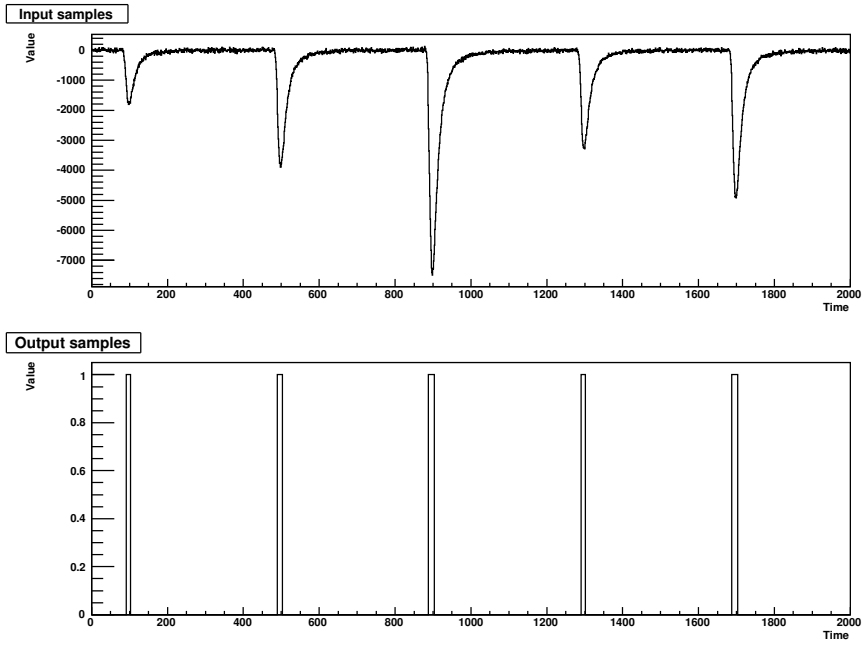

Figure 8: Simulated Landau pulses as input with output timing pulses generated by a leading edge discriminator.

COTS hardware-based CFDs tend to be expensive in price, power consumption, and rack space (e.g. Ortec NIM module [7]). The performance of an FPGA-based CFD implementation will of course be limited by ADC sample-rates, so this should be considered when making direct comparisons between this example and dedicated hardware implementations.

This software implementation of a CFD is similar to the method used in an analog hardware based CFD. A key requirement for a CFD is that the rise-time of the input pulses must be known. An input waveform is inverted and delayed (by the rise-time) before being added to an attenuated (by a factor of 4) version of the original input waveform. These operations generate a bipolar waveform with a zero crossing at the time the original pulse reaches 25% of maximum.

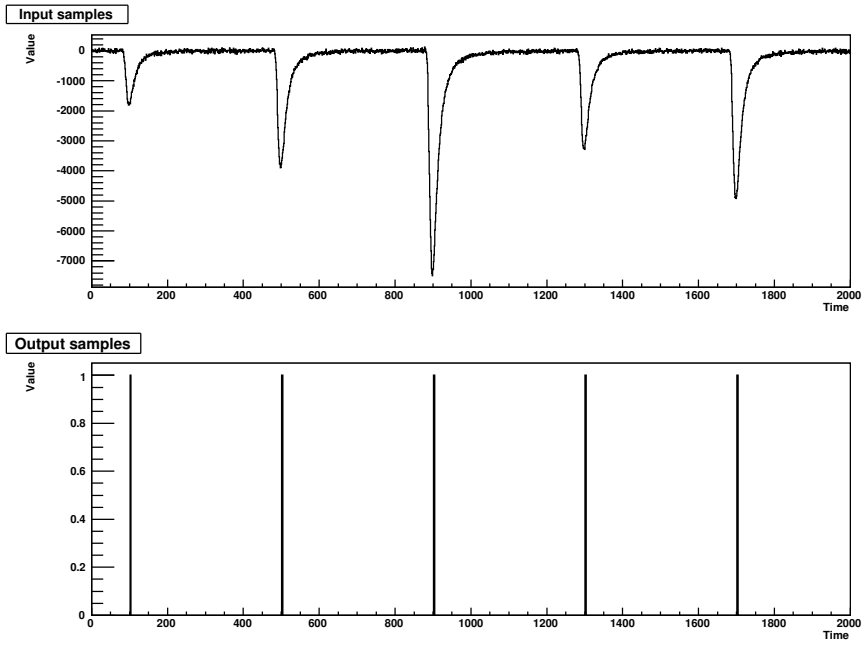

Figure 9: Simulated Landau pulses as input with output timing pulses generated by a constant fraction discriminator.

Comparison of time walk for constant fraction and leading edge discriminators A large number of pulses with varied heights were analyzed using both leading edge and constant fraction discriminators. Time walk is histogrammed in Figure 10 and Figure 11 for each type of timing discriminator. The symbol, T_{MPV}, in the time walk figures is the time corresponding to the peak of the generated Landau pulse.

3.4 Digital pulse shaping examples

Analysis of electrical pulses is of critical importance to the extraction of valuable information from many types of detectors. Digital pulse shaping is a type of digital pulse analysis that is optimized for streaming processing of waveforms. Many analog filtering techniques can be applied in the digital realm to sampled waveforms. The two following pulse shaping

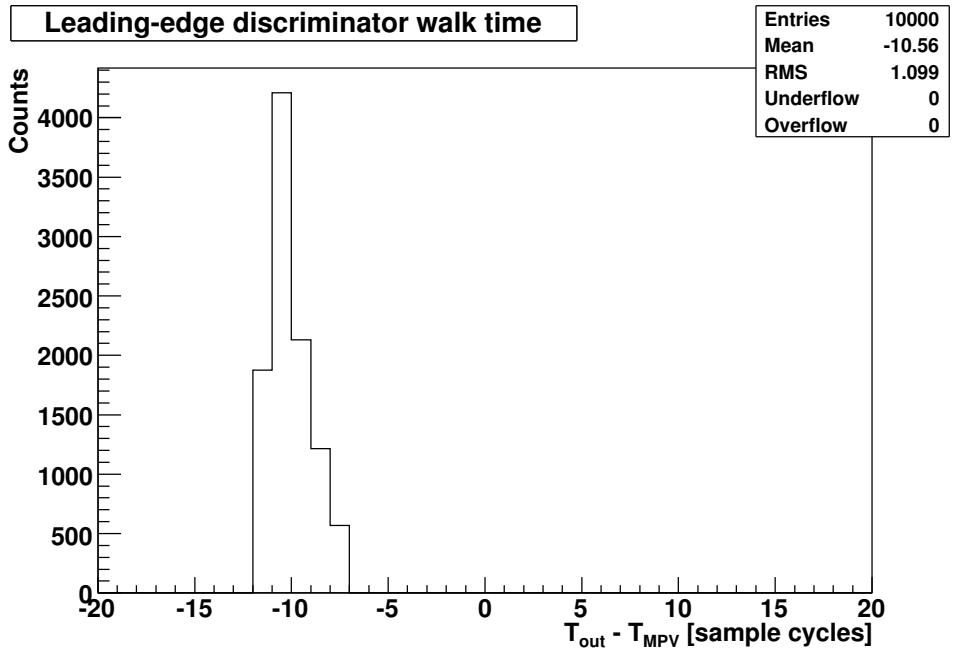

Figure 10: Systematically reduced timing resolution from a leading edge discriminator.

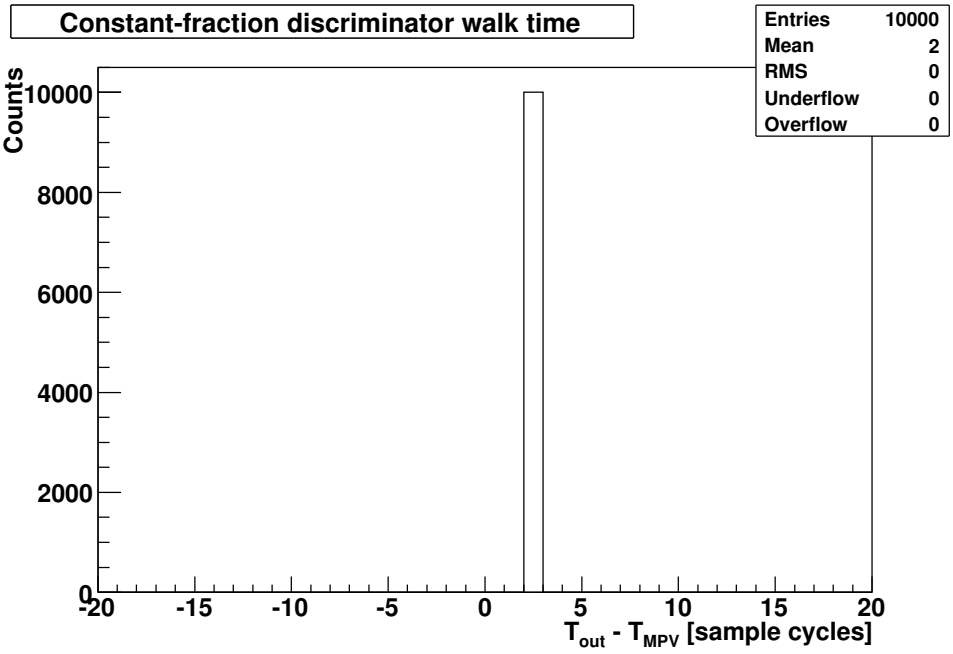

Figure 11: Single ADC clock-cycle precision timing resolution from a constant fraction discriminator.

examples are used in spectroscopy systems (e.g. gamma ray). As it is more and more feasible to put ADCs on *every channel* and process digital samples on front-end FPGAs, similar techniques become interesting for other detector applications.

3.4.1 Trapezoidal pulse shaper for exponential pulses

This pulse shaping example is an implementation of an algorithm given in Reference [8]. The trapezoidal pulse shaper has the effect of generating a trapezoid (or triangular) pulse from an input exponential pulse.

In Figure Figure 12 the input pulse was the exponential (black) curve and the output of the FPGA pulse shaping algorithm is the trapezoidal (red) pulse. The code for this is very short and can easily process 100 MS/sec without losing any of the samples. A more realistic case would involve noise being present in the signal. In Figure Figure 13 the same shaping is applied with a noisy exponential input pulse. The zero line here has been raised so that the baseline noise can be seen. It is clearly seen in this figure that the noise in the output trapezoidal pulse is significantly smaller than the noise in the input pulse. Thus not only is it easy to produce a pulse of different shape than the input pulse, but it is also trivial to reduce noise in the pulse at the same time.

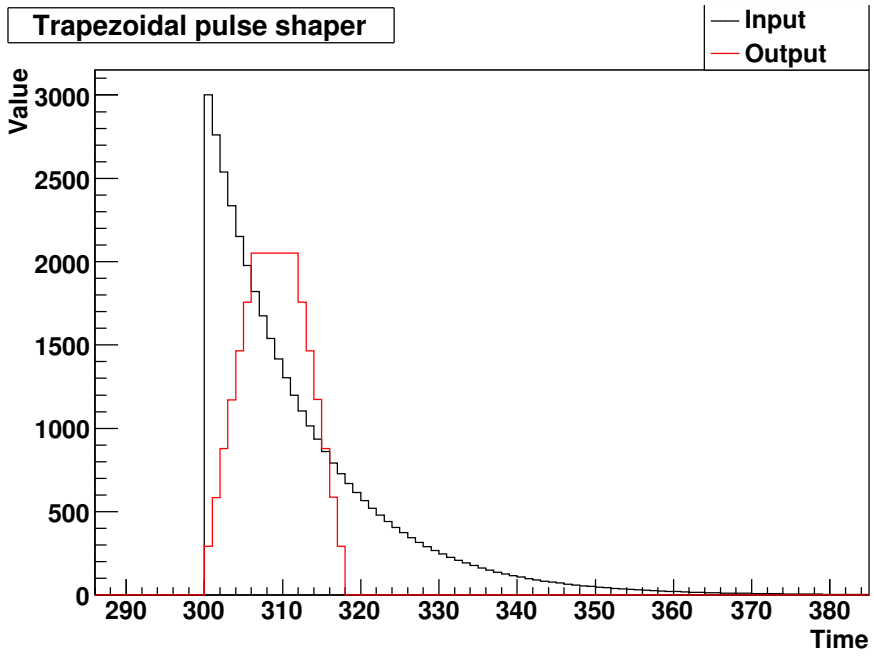

Figure 12: Simulated sampled exponential pulse and result after application of the digital trapezoidal shaper.

The same basic principles are shown in producing a cusp-like pulse as shown in Figures Figure 14 and Figure 15. The cusp-like filter is also an implementation of an algorithm presented in Reference [8]. The second case once again adds noise to the exponential input pulse, clearly showing the reduction in noise while outputting a pulse shape of whatever

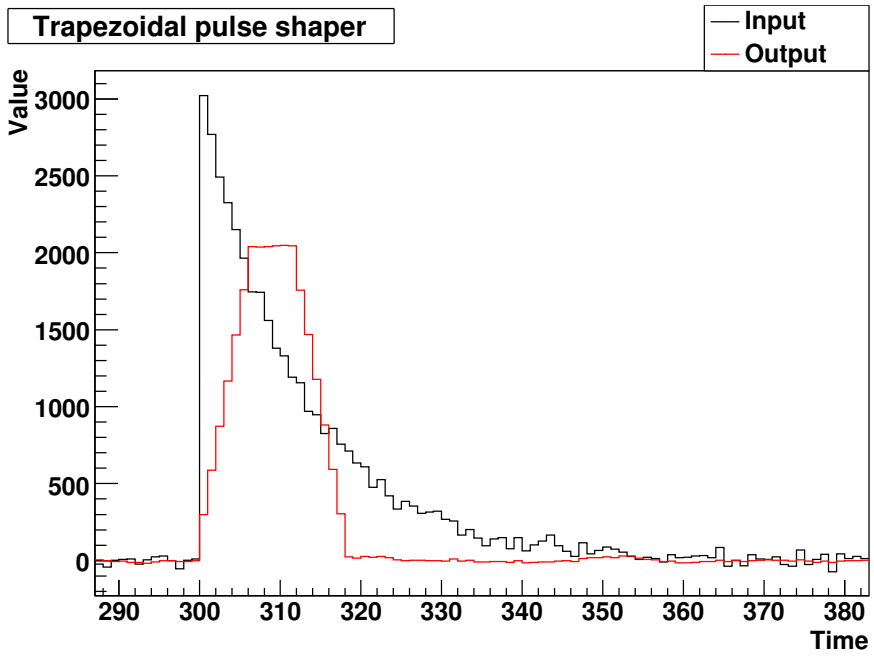

Figure 13: Simulated sampled exponential pulse with noise added and result after application of the digital trapezoidal shaper.

type is needed. These are fairly simple examples, but they demonstrate that digital pulse shaping FPGA code can be produced efficiently and flexibly.

3.5 Principal component analysis

Component analysis techniques are generally used to reduce high-dimensional data to lower-dimensional representations. Principal component analysis (PCA) is a linear method to accomplish this task. In a physical system, PCA can often be used to extract dimensions that are most significant physically (i.e. separating interesting signals from noise or combining redundant measurements).

During Phase I, an example PCA was developed based on candidate tracks in a simplified version of the Fermilab E906 di-muon spectrometer. Geant4 was used to simulate particle tracks through the detector system. Figure 16 shows the detector geometry and simulated tracks. The basic premise of this example is the common generic PCA process might be used as a pre-processing filter in part of an advanced FPGA accelerated trigger system.

Since the plan for Phase II is to work with Fermilab E906 this process was tested on taking measurements made at x-y hodoscope planes to assess the ability to find tracks in real-time. In this test six x-y planes were used, for a total of twelve measurements. E906 will likely only have four x-y planes, but it was decided to increase the number of planes to better demonstrate the performance of the method. The basic layout of E906 is shown in Figure 16, which is a two-magnet spectrometer. This experiment is a followup experiment to Fermilab E866/NuSea, which determined the ratio of anti-down to anti-up quarks over an approximate Bjorken x range of 0.05 to 0.25 via the Drell-Yan process. The DY process is

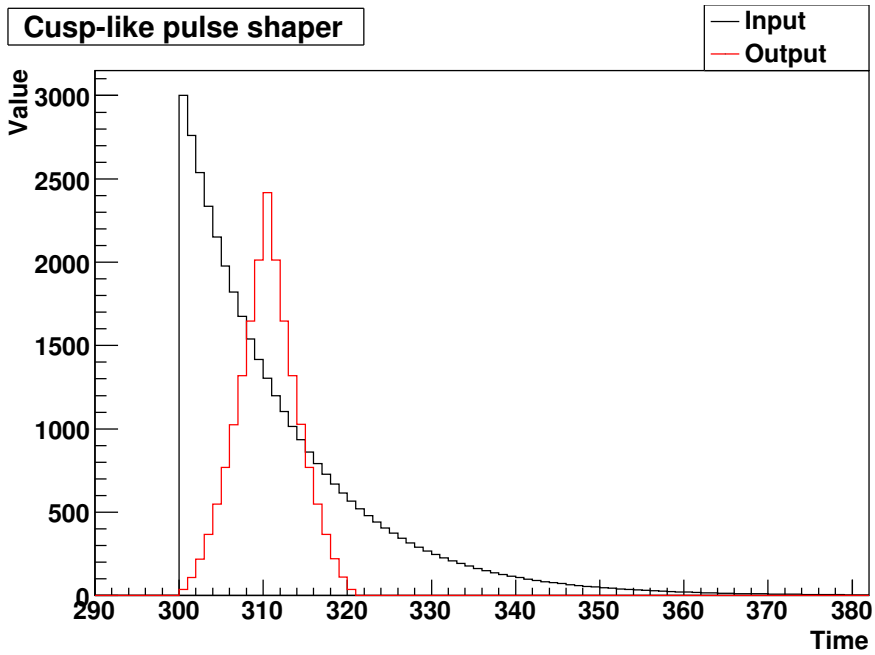

Figure 14: Simulated sampled exponential pulse and result after application of the digital cusp-like shaper.

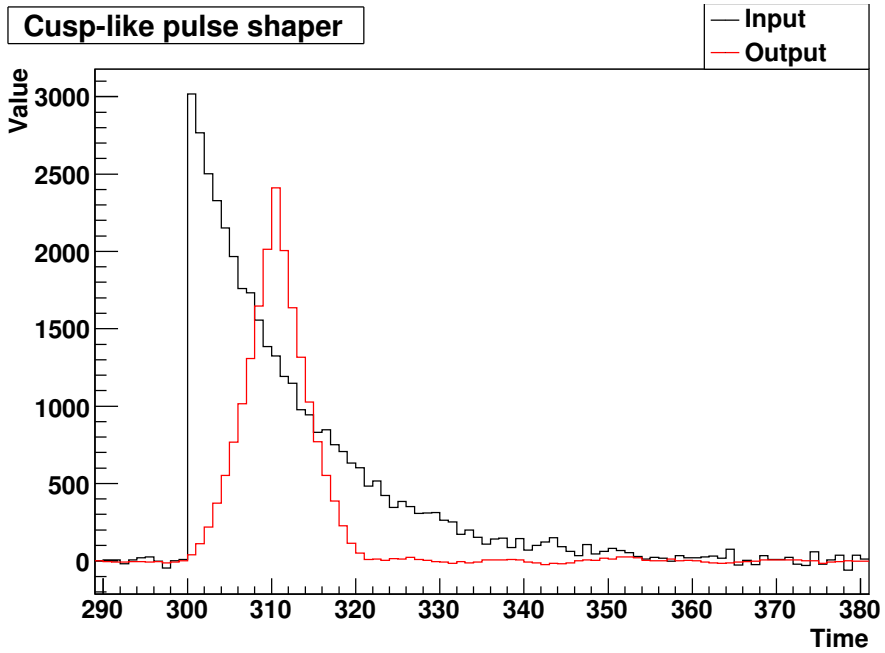

Figure 15: Simulated sampled exponential pulse with noise and result after application of the digital cusp-like shaper.

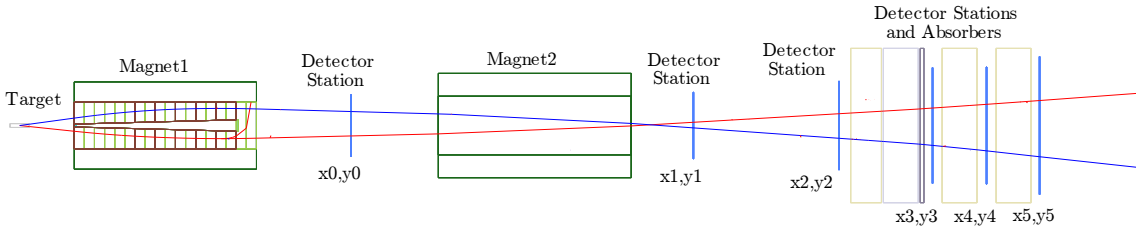

Figure 16: Simplified E906 di-muon spectrometer geometry from Geant4 simulation. The positive muon track is colored blue. Negative particle tracks (negative muon and ionization electrons) are colored red. For simplicity, this simulation assumes x-y coordinates will be recorded at all six detector planes, while the actual experiment has some y-only planes. The length of the apparatus is approximately 26 meters.

when a quark in the beam proton annihilates an anti-quark in the target proton or neutron sea, producing a virtual photon which then decays into a pair of leptons. The easiest of course to detect is a di-muon pair due to high energy muons ability to penetrate material used to absorb hadrons. E906 will extend the Nu/Sea measurements up to an x of around 0.5.

Figure 17 shows the distribution of input data for the PCA. These data hit coordinates at detector planes valid simulated positive muon tracks. Figure 18 shows the same dataset after it has been transformed into the PCA coordinate space. The signifcant coordinates in this case (pc0, pc1, pc2) correspond to a scaled linear combination of each muon's initial momentum vector. The outliers in the higher coordinates of Figure 18 correspond to muons that have undergone significant multiple scattering.

Invalid tracks (not shown) would tend to be very large in the higher PCA transformed coordinates, while valid tracks would be generally very small in the higher PCA transformed coordinates. This information could be used as the basis for a naïve track classifier, or it could be combined with other data or more complicated techniques (e.g. artificial neural networks) to create a powerful track classifier.

The reason this result is extremely interesting is that one could perform a PCA on input data from a detector to determine if a particle track is present. Because of the parallel nature of the FPGA and its inherent pipeline structure, one gets a PCA result on every clock cycle. This makes it possible to consider implementing a PCA hardware trigger, which would allow one to check for possible tracks while throwing none of the data away. One of the goals of working on FPGAs for the trigger hodoscopes for E906 to test this methodology is to produce a trigger with zero dead time. This would mean that while doing a higher level track reconstruction process, as compared to a set of lookup matrices, no data are lost if the event being processed turns out to either have no tracks or has too many tracks to successfully reconstruct. In the case of E906 the problem appears to be very doable since the experiment is looking for di-muon pairs from Drell-Yan interactions. This simplifies the type of events one wants to record significantly, so Innovation Partners will work with the E906 collaboration to produce the desired FPGA code for the trigger to run on FPGA boards based on designs already existing for other parts of the E906 detector. The Principal Investigator has already worked in close contact with Pat McGaughey of LANL

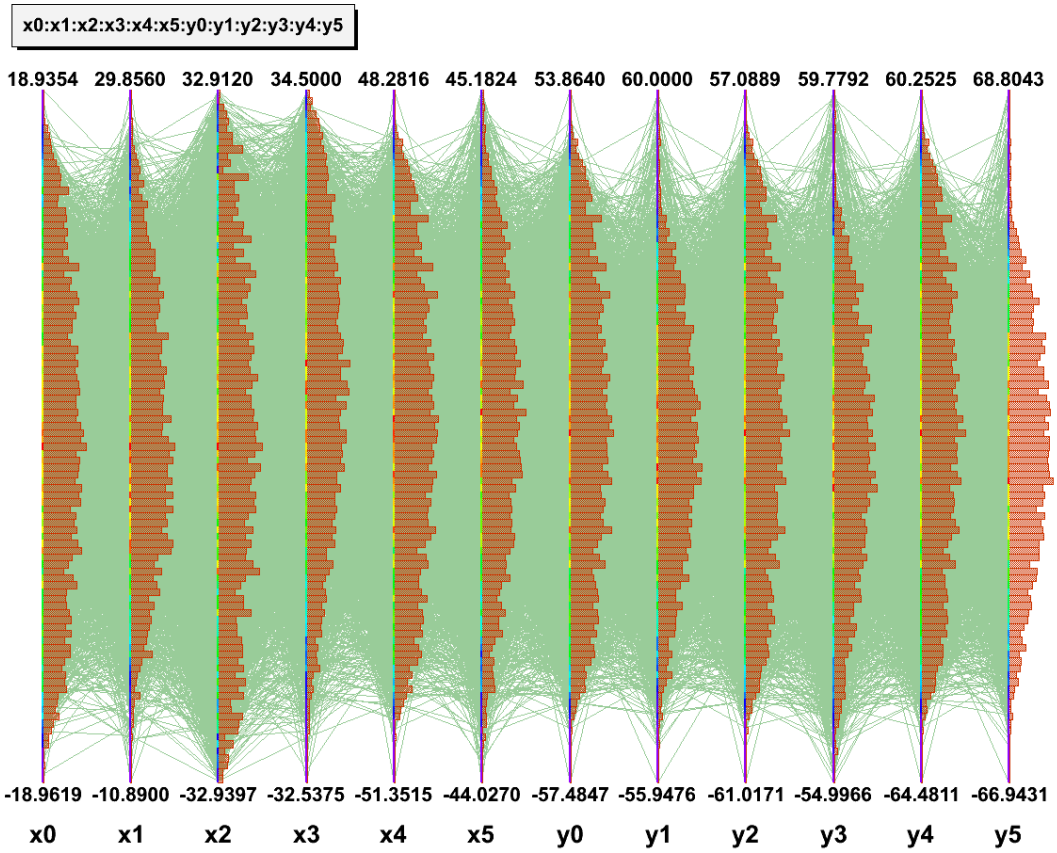

Figure 17: Parallel coordinate diagram showing the distribution of input x and y coordinates in the laboratory coordinate system for valid positive muon tracks. These values are measured in inches at each detector plane.

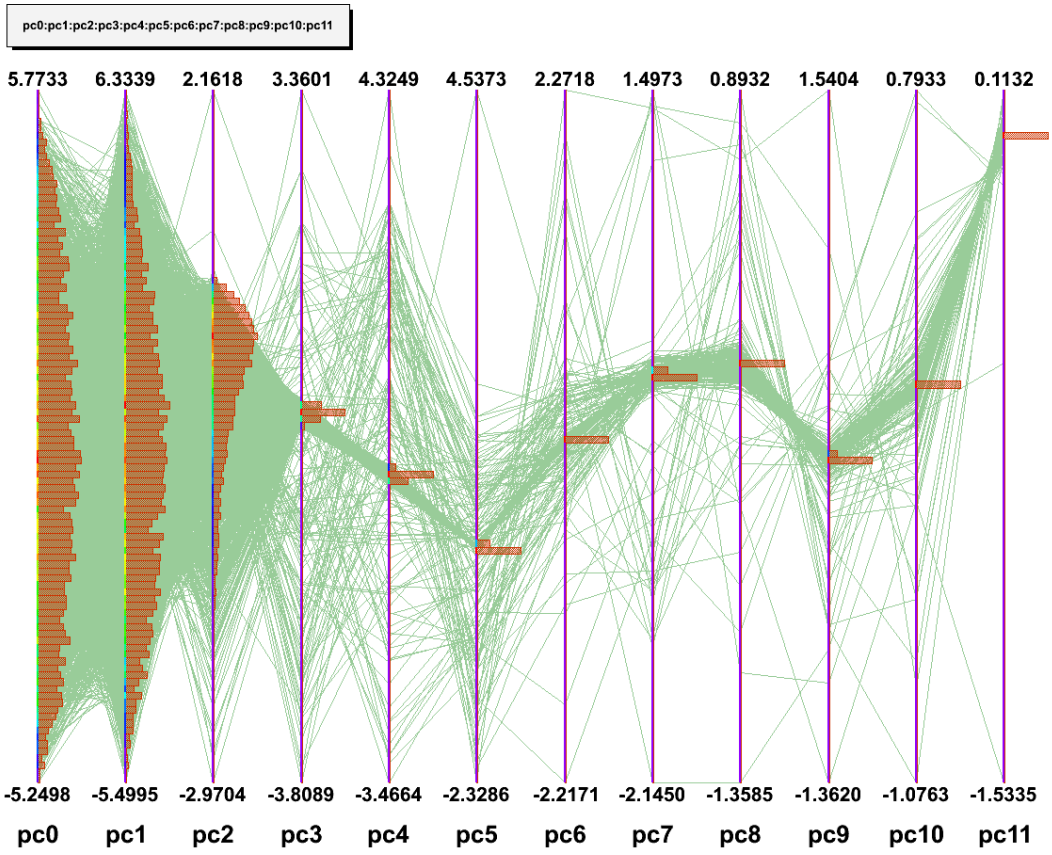

Figure 18: Parallel coordinate diagram showing the distribution of valid positive muon tracks that have been transformed into the principal coordinate space. The high order coordinates (pc9, pc10, pc11) are almost negligible for most valid tracks as most physical information is in the lowest three coordinates (pc0, pc1, pc2).