

Trigger and DAQ for $K_L \rightarrow \pi^0 \nu \bar{\nu}$ Experiment at J-Parc

Anton Kapliy

*Enrico Fermi Institute, The University of Chicago
5640 South Ellis Avenue, Chicago, Illinois 60637, USA*

May 15, 2007

Abstract

We propose a design of Level-1 trigger and readout chain for the upcoming J-Parc experiment that supports trigger rates in excess of 100 KHz with virtually no downtime. The design has been implemented on an Altera Stratix II FPGA Development Kit. A few preliminary tests suggest that the FPGA satisfies the timing requirements and has enough memory to service 16 readout channels.

1 Experiment overview

1.1 Physics motivation

The very rare decay $K_L \rightarrow \pi^0 \nu \bar{\nu}$ provides one of the best probes for understanding the origin of CP violation in the quark sector. The decay is a Flavor Changing Neutral Current (FCNC) process from strange quark to down quark (see Fig. 1), and its observation will provide new positive evidence for direct CP violation. The ultimate goal of the experiment is to achieve a sensitivity of better than 3×10^{-13} for the branching ratio, corresponding to an observation of more than 100 events for the Standard Model prediction[1].

The $K_L \rightarrow \pi^0 \nu \bar{\nu}$ decay is a theoretically clean decay mode, but also a difficult decay mode to study experimentally. The incoming K_L cannot be directly observed before decaying, and only two photons are visible. The branching ratio expected from Standard Model is very low, $\simeq 3 \times 10^{-11}$, whereas one of the major backgrounds, $K_L \rightarrow \pi^0 \pi^0$ (with two photons missed) has the branching ratio of 8.83×10^{-4} [1]. Thus, the background must be suppressed by 8 orders of magnitude or more.

1.2 Detector

The experiment uses a collimated K_L beam produced in the collisions of protons with a stationary target. The main features of the detector (see Fig. 2) are:

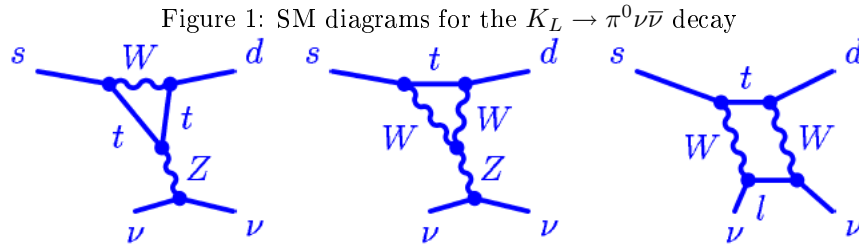
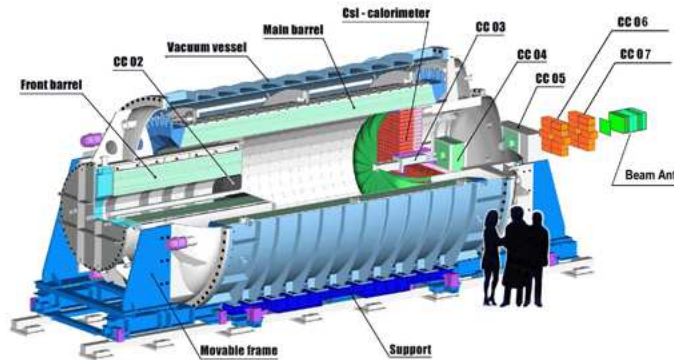


Figure 2: KEK PS E391a detector



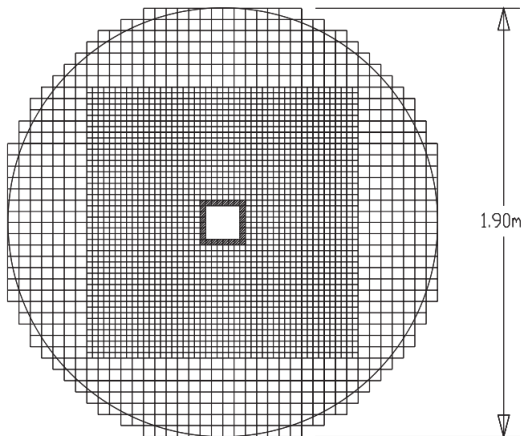
- Electromagnetic (EM) CsI calorimeter - placed downstream of the decay region to catch two photons from the $K_L \rightarrow \pi^0 \nu \bar{\nu}$ decay.
- Hermetic photon veto counters - cover the decay region to suppress the $K_L \rightarrow \pi^0 \pi^0$ backgrounds by detecting the two extra photons.
- High Vacuum - the decay volume is evacuated to suppress neutrino interactions with residual gas.
- Narrow K_L beam - reduces the error on the transverse momentum of π^0 .

The primary decay mode of neutral pions is $\pi^0 \rightarrow \gamma\gamma$, with the branching ratio of 98.8%[2]. Therefore, one must develop efficient techniques to detect final-state photons.

1.3 EM calorimeter

The EM calorimeter measures the positions and energies of photons to reconstruct π^0 in the $K_L \rightarrow \pi^0 \nu \bar{\nu}$ decay and discriminate against extra photons in the $K_L \rightarrow \pi^0 \pi^0$ decay. The calorimeter consists of 2816 pure CsI crystals spanning a region of 50 cm along the beam axis and nearly 2 meters in the radial direction

Figure 3: EM Calorimeter, cross-sectional view. The inner crystals are more finely spaced to give better resolution in the fiducial region.



(see Fig. 3). Each crystal is connected to a photomultiplier tube (PMT) that converts the energy of scintillated light into a measurable signal.

1.4 Analog electronics

The input pulse from PMT has a rise time of about $5ns$ and a slowly falling edge. To facilitate digitization and reconstruction, this pulse is shaped with a 7-pole Bessel filter that gives it a quasi-gaussian shape with a fixed FWHM of $45ns$ [3]. The filtered output is sampled by a 14-bit, $125MHz$ Flash ADC¹.

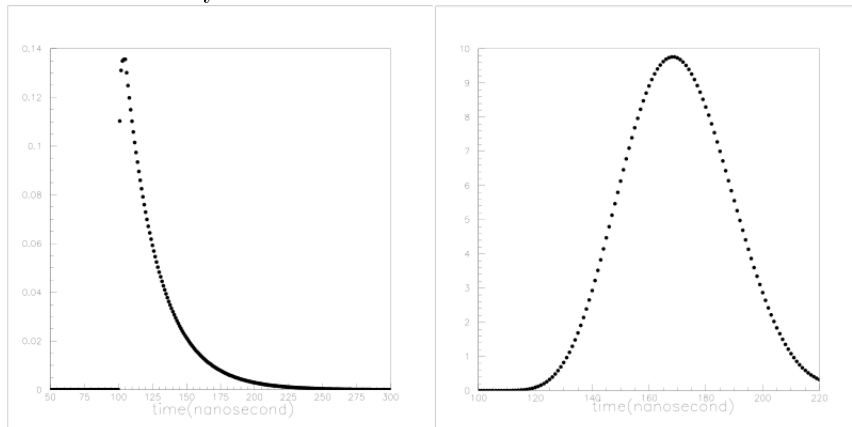
1.5 Data extraction

Given digitized output of the Bessel filter, one can retrieve the information about the signal by fitting the sampling points with a gaussian function using the chi-square method. The fitted peak identifies the time of the hit, and the amplitude parameter gives the charge from PMT.

The fitting results depend on when we sample the pulse, i.e. on the sampling starting points. This effect, along with other systematics, has been extensively studied. See, for example, [3]. The obtained energy resolution is less than 2%, which is largely due to asynchronous sampling. Timing resolution is better than $0.5ns$ when energy deposited is above $10MeV$, and the two pulses can be separated if they are more than $20ns$ apart. This is quite remarkable, given that the points are only sampled every $8ns$.

¹A Flash ADC is a fast digital-to-analog converter that uses a linear voltage ladder to compare input voltage to successive reference voltages. The output of these comparators is fed into a digital encoder that converts its input into a binary value.

Figure 4: Approximated pulse from PMT, and the Bessel filter shaped pulse. y-axis is in arbitrary units.



2 Digital electronics

2.1 Overview

It is impractical to save all the data as a steady stream of words digitized every $8ns$. A simple calculation shows that 2816 channels sampled as 14-bit words at $125MHz$ give a data rate of about 570 gigabytes/second. Even setting aside the issue of transferring so much data from Flash ADC's to a computer, storing it would require a prohibitively expensive IT infrastructure.

Luckily, we are only interested in a portion of this data that contains photon hits. By applying a predefined energy threshold right after digitization², one can filter out irrelevant signal and only save the few 14-bit words deemed acceptable by the trigger algorithm. In fact, extrapolation from the predecessor experiment, KEK PS E391a, shows that the average photon hit rate is around $100KHz$ for the entire detector[4]. Assuming that for each hit we save 30 samples spaced in $8ns$ intervals (a $240ns$ window), the data rate comes down to a feasible 5 megabytes/second.

Digital processing of the Bessel filter pulses is done in an Altera Stratix II FPGA³ chip. Each chip, capable of servicing 16 CsI calorimeter crystals, is installed on a custom-designed DAQ board. 16 such boards, in their turn, plug into a VME⁴ crate that interfaces to a computer.

Each FPGA provides the following functionality:

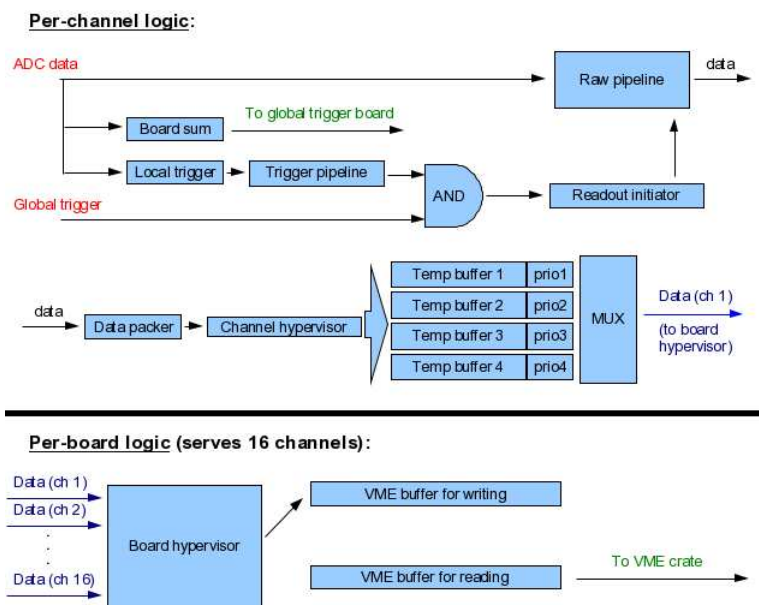
- Data pipelining

²This procedure is called *triggering*.

³A Field Programmable Gate Array (FPGA) is a specially made digital semiconductor device that utilizes gate array technology and can be reprogrammed after it is manufactured.

⁴Versa-Module Eurocard (VME) is a standard computer bus definition, providing high-speed data transfer

Figure 5: Digital logic in the Altera FPGA



- Local triggering
- Computation of “board-sum“ - energy of all 16 channels serviced by a chip
- Transmission of “board-sum“ to a global trigger board
- Data packing
- Interface to VME readout

This paper reports on a successful implementation of these algorithms and some preliminary field tests.

2.2 Block diagram of digital logic

Figure 5 shows a block diagram of digital logic programmed into an FPGA. The top part of the figure contains logic that is repeated 16 times - once for each channel. The bottom part aggregates this data into a single buffer available for computer readout. Individual blocks will be discussed in detail in the subsequent sections.

2.3 Trigger

As the ADC data comes in, a copy of it is saved into a pipeline of variable length to await the trigger decision (“raw pipeline“ in Fig. 5). A channel is considered

triggered when there is a coincidence between local trigger and global trigger.

Some corrections are applied to raw data before trigger, such as pedestal subtraction and averaging. Note, however, that these corrections are only used for triggering, and final readout contains raw data.

2.3.1 Pedestal subtraction

Analog electronics and the flash ADC may introduce a small baseline that can be corrected by means of pedestal subtraction. Pedestal corresponds to the ADC output averaged over 16 clock cycles in the absence of physics signal. Its value is recomputed every 3 seconds while the detector is idle⁵ and then subtracted from all subsequent data.

2.3.2 Averaging

To prevent false triggering due to isolated voltage spikes, we compute a running average of the signal over 4 clock cycles. This effectively smoothens out the signal profile and removes jitter.

Figure 6 shows pedestal subtraction and averaging applied to a sample pulse. The pulse has correct overall shape, but its baseline level and noise jitter were over-emphasized to better illustrate the algorithm. The digits were obtained in dedicated Altera simulation and were also checked in an actual FPGA chip.

2.3.3 Local trigger

After pedestal subtraction and averaging the signal is compared with a threshold value. Each channel has its individual threshold that can be programmed through VME registers. If a threshold is exceeded, a high bit is placed into a local threshold pipeline (see Fig. 5) and waits there for about $2\mu s$, until global trigger arrives. The two are aligned in time by setting the proper pipeline depth.

2.3.4 Global trigger and board-sum

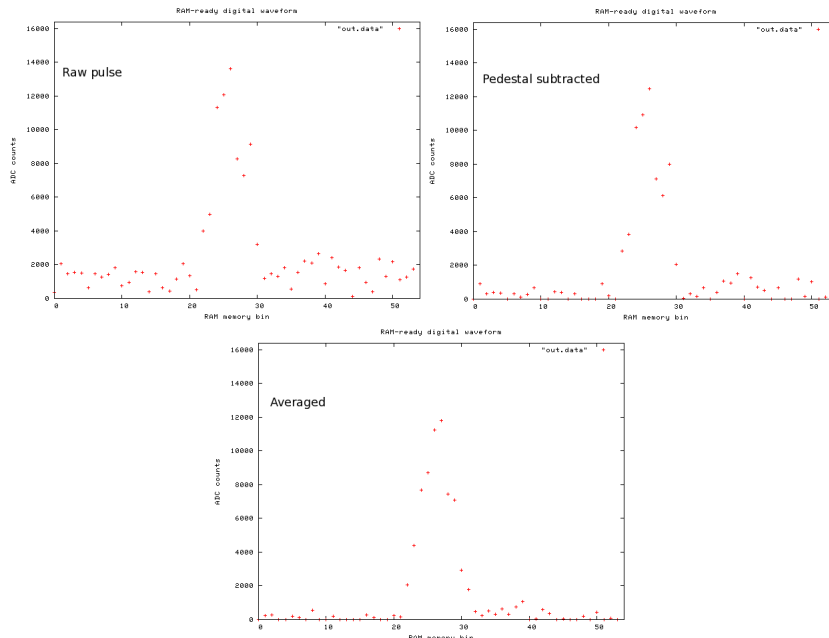
Global trigger comes from a trigger board and signifies whether the event as a whole is interesting. While the exact logic behind global trigger will not be finalized until a trigger board is designed, it will be based on the board-sum energies reported by every DAQ board.

The board-sum energy (pedestal-subtracted and averaged) is computed in the FPGA using a ladder of registered adders. The result, a sum of 16 channels, is truncated into two 6-bit words and sent out at $750MHz$ via LVDS SERDES⁶ to the trigger board.

⁵Interactions occur in 1-second intervals separated by 2 seconds of idle time

⁶A SERializer/DESerializer (SERDES) is a device used in high speed communications. It converts data from a parallel data stream to a serial data stream.

Figure 6: Illustration of pedestal subtraction and averaging



2.4 Readout

When a coincidence between local and global triggers is established, the pulse that gave rise to this trigger is stored somewhere within the $2\mu s$ raw pipeline (see Fig. 5). By properly setting its depth, implemented as an adjustable delay parameter, we can arrange for this data to come out in time with the trigger.

In reality, we do not want to save just one word that was triggered, but the entire pulse. For this reason, we take out 32 words centered around the trigger point, covering a $256ns$ interval. The raw pipeline delay is configured so that the peak of the pulse is located around the 16^{th} word.

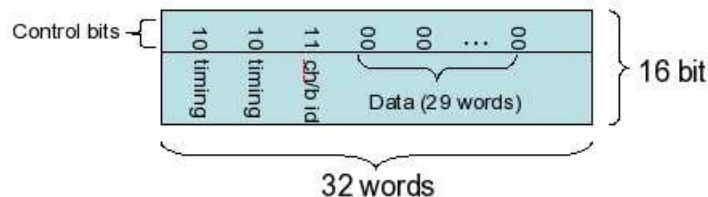
2.4.1 VME-based readout

Ultimately, the data must be transferred to a computer through a VME bus. Since negotiating a transfer can be timely, the data is collected in a large buffer inside the FPGA and then transferred at once. The size of the buffer is limited by the amount of free memory available on the device, which is about 32,000 32-bit words for Altera Stratix II FPGA.

We propose to have a spill every 10 milliseconds. This gives a wide margin of safety as far as memory is concerned: assuming a per-channel rate of $\sim 100KHz$ (a gross overestimate, since this figure refers to the expected *global* rate for all 2816 channels), we'd only need 16,000 32-bit words per data spill.

We provide two large memory buffers, so that at any given time one is used

Figure 7: Data packer showing timing, channel/board ID, and data



for writing new data, while the other is read out by VME master controller. On every spill these buffers are interchanged, which ensures continuous operation of the system.

2.4.2 Data packer

An event takes 32 14-bit words and requires some identification to tell it apart from other events. This information is tagged to the first 3 words, leaving 29 words for the actual data. An integer between 0 and 1.25M is used to identify an event within a 1/100 second spill. A number from 0 to 15 identifies an event within the sixteen channels on the DAQ board. Finally, the DAQ board ID is used to identify the DAQ board. See Figure 7.

2.4.3 Temporary buffers

Unfortunately, it is often impossible to immediately place an entire event (32 words) into the final VME buffer. Several channels can trigger simultaneously, but the two-port VME memory only allows one write operation per clock. The first channel may start writing immediately, but others must be pipelined until VME buffer is ready to accept new events (or we will introduce unnecessary downtime).

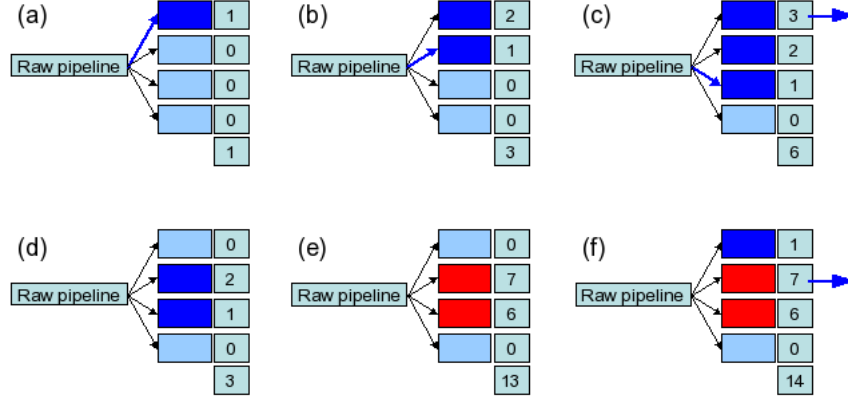
We introduce four temporary buffers per channel that can store an entire 32-word event. These buffers save triggered events and feed them into the VME buffer only when it's ready to accept new data, ensuring that no events are lost. The following algorithm was developed to optimize the routing between temporary buffers and the VME-addressable buffer. The algorithm derives from the author's experience with Linux Kernel scheduler.

2.4.4 Channel hypervisor

Each temporary buffer is assigned a priority value, implemented as an array of D flip-flops. A **Channel hypervisor** (see Fig. 5) is placed in front of the four buffers to route incoming events and manage priorities.

The priority system is illustrated in Figure 8. It shows four buffers with corresponding priorities, as well as the total priority for the channel. In (a), an event is saved into the first buffer and gets priority 1. In (b), a new event is

Figure 8: Priority system management by Channel hypervisor



saved into the second buffer with priority 1. At the same time, the first buffer has its priority incremented by 1 unit, because it contains an *earlier event* that should have higher priority. In (c), even another event comes and is saved into the third buffer, while the first and second priorities are incremented to reflect their time precedence⁷. A blue arrow next to the first buffer in (c) signifies that the VME memory is finally able to accept an event and selects the first buffer (which has the highest priority) for readout. The result of this action is in (d), where the first buffer has been read out and its priority is zeroed.

In fact, VME memory doesn't have to wait until an event is fully saved in a temporary buffer, as was shown in Fig. 8. VME buffer will start saving an event almost immediately after it enters a temporary buffer, thus reducing latency by ~ 30 clock cycles. Similarly, the temporary buffer becomes available for new data almost immediately after a (temporary buffer \rightarrow VME buffer) transfer is initiated, further improving utilization of available memory.

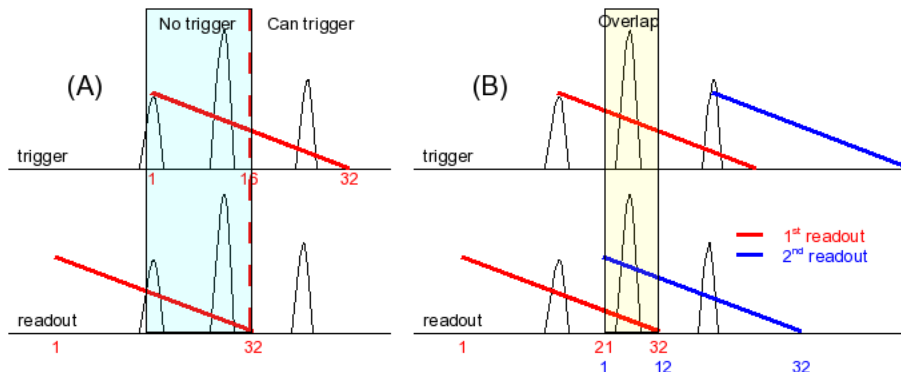
2.4.5 Trigger blocking and second readout chain

If there is an actual photon hit in the detector, the digitized pulse would exceed trigger threshold over a few clock cycles. Since we want only one trigger per photon hit, further triggering is blocked for 16 clock cycles. The choice of this number is justified in Figure 9.

Referring to part (A) of the figure, suppose there are three pulses that exceed trigger threshold. The first two are separated by less than 16 clock cycles, while the third is over 16 clocks away. The top bar shows trigger comparator marching over 32 clock cycles of data. When the first pulse gives a trigger, we immediately route the data into a temporary buffer, starting 16 clocks earlier. This positions the pulse in the middle of a 32-word event readout (see bottom of part (A))

⁷Maximum priority is set at 4, because data should not wait more than 4 event cycles (256 ns) before moving to the VME buffer. If this happens, a pile-up error bit is reported, suggesting that the threshold is raised to decrease trigger rate.

Figure 9: Trigger blocking and the second readout chain



in Fig. 9). Incidentally, the second pulse is also saved in the same event, so it doesn't require a separate trigger. This allows us to block triggering for 16 clock cycles, indicated by blue shading in the figure.

In part (B), a third pulse is already beyond the "no-trigger" region. Therefore, this event is transferred into another temporary buffer, as shown by the blue line. Since the second event also starts 16 clocks earlier (to position the pulse in the middle of a 32-word readout), there is a small overlap between two events. Incidentally, the second pulse is saved twice, as shown by light-yellow shading in part (B).

Such an overlap implies that occasionally data from the "raw pipeline" (see Fig. 5) must be simultaneously routed to two temporary buffers. This dual readout chain was fully incorporated into the priority scheme described in section 2.4.4.

2.4.6 Board hypervisor

Each of 16 channels reports its total priority (sum of four priorities) to a board hypervisor. The board hypervisor selects the highest-priority channel and transfers its highest-priority event into the final VME buffer. It also keeps track of which VME buffer is being used for writing new data, and which one is being read out by a computer (see Fig. 5).

2.4.7 Continuity between spills

A small caveat pointed out by Harold Sanders is that in the end of every 10 ms, a spill is requested and we must switch the VME buffer into computer readout mode. However, this switch cannot occur immediately if there is data from a previous spill in the temporary buffers.

This is remedied by delaying VME buffer switching until all "dirty" buffers are transferred to the VME buffer. Referring back to Fig. 8(e), a spill request makes the two filled buffers critical by incrementing their priorities by 5 units.

Since normal priorities are limited to 4 units, critical buffers are guaranteed to be read out first. In part (f), an event belonging to a new spill is saved into the first buffer with priority 1, and it won't be read out until all critical data is transferred to the VME buffer.

3 Implementation

All logic described in this paper⁸ was designed, implemented, timed together, and tested using a Stratix II DSP Development Kit[5]. The development kit features the same FPGA chip that will be used in the actual experiment. It includes simple ADC and DAC that allow to take external signals from a pulse generator or output internal digital signals to the oscilloscope. It also has four programmable push-buttons and a few LEDs that allow simple user interaction.

Most of the work was done in Altera's Integrated Development Environment, Quartus II, Version 6.1[6]. Basic blocks were designed using graphical block editor, while more elaborate elements, such as hypervisors and data routers, were written in proprietary Altera's Hardware Description Language (AHDL). The firmware was uploaded to the chip through a JTAG interface[7].

3.1 Test pulse

Digital test pulses were modeled after Jiansen Ma's simulations and integrated into the chip. Extra jitter was added to easier identify the pulses on scope traces.

While multiple tests were conducted to ensure data integrity and algorithm robustness⁹, here we report on a particular test. We send two pulses separated by a few clock cycles into channel 1 in order to create a situation similar to Figure 9-B. This activates dual readout and allows us to observe common overlap between two events. At the same time, a single pulse is sent into channel 2 in order to test the performance of temporary buffers and hypervisors, as all three pulses cannot be transferred to the VME buffer simultaneously. The pulses are dispatched when the user presses a push-button on the board. As the board is always on stand-by, it self triggers¹⁰ the pulse, formats it, and saves into a final VME buffer. When the user presses another button (simulating a spill request), the VME buffers are switched, and the contents of the last used VME buffer are sent to a DAC to visualize all collected data.

Referring to Figure 10:

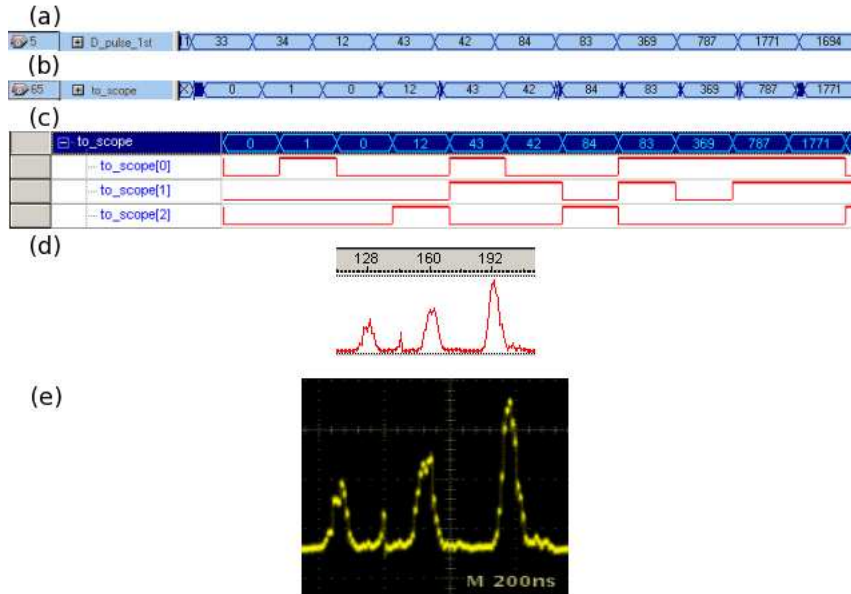
1. (a) shows the digits that form a raising edge of the first pulse sent to channel 1. Each block represent an ADC count value with 8 ns separation.

⁸with the exception of LVDS SERDES, due to hardware limitations

⁹A reader is welcome to stop by University of Chicago's Electronics Shop and witness the system first-hand

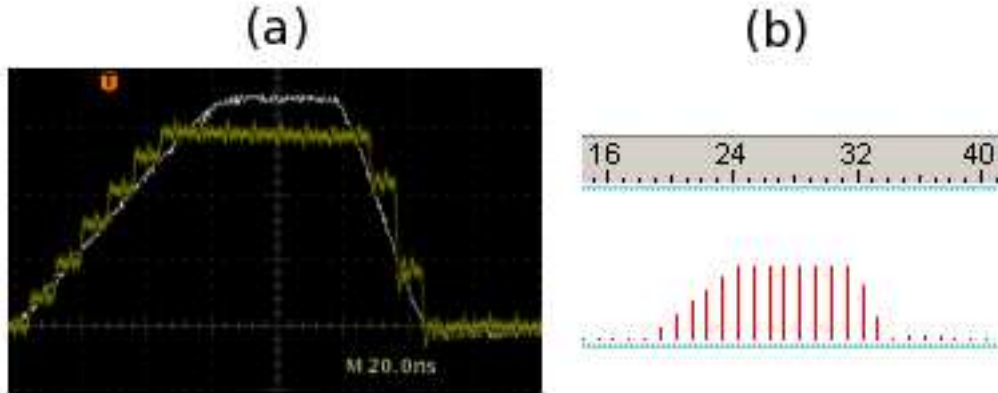
¹⁰Global trigger was disabled for this test, because trigger board has not been designed yet

Figure 10: Test pulse run



2. (b) shows part of the formatted event stored in a VME buffer. It was obtained in a dedicated Altera simulation that takes into account propagation delays in the chip (thus, jitter on clock transitions). The 0-1-0 pattern is a header that identifies channel number. Event ID and board ID were not tagged to the event for purposes of simplicity. The data integrity is preserved compared to (a), and data collection starts several words before the peak of the pulse. Note that that the entire readout was checked word-by-word and shown to be as expected.
3. (c) is the same as (b), but it was read out from an actual chip through a SignalTap debugging node[8].
4. (d) expands the range of (c) and shows all three formatted events saved into VME buffer. The digital words are plotted on a graph and connected by a smooth line. Horizontal axis labels data words, each of which corresponds to an 8-ns interval. The little spike in the middle is the raising edge of the second pulse that made its way into the first event, similar to Figure 9-B. Following are the second pulse sent to channel 1, and the pulse sent to channel 2. Channel numbers tagged to the event header cannot be seen on the graph, but were checked to be correct.
5. (e) are the three saved events sent to the scope via on-board DAC.

Figure 11: External analog pulse



3.2 External analog pulse

The test board is also able to accept external analog pulses through an on-board ADC. It self-triggers on the incoming data and saves formatted events into a VME buffer.

For purposes of demonstration, an extra wide pulse with different leading and trailing edges was sent asynchronously from a LeCroy pulse generator. In Figure 11-a, the white line is the original pulse from LeCroy, and the yellow line is the event saved into a VME buffer and returned to the scope through on-board DAC. (b) shows the same event before analog-to-digital conversion.

3.3 Robustness

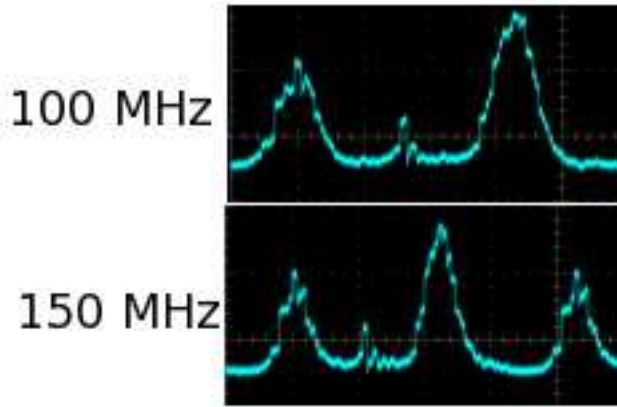
At early stages of design, we encountered a problem when the FPGA couldn't keep up with the 125 MHz clock. Some bits occasionally fell out due to slow signal propagation inside the chip. After a few revisions the problem was fixed, but the question remains whether the design is robust, or barely works at the required clock speed.

To test robustness of the design, a Phase-Locked Loop (PLL)[9] was used to generate several clocks. The design was run on the development board at clock speeds of 100, 125, 150, and 200 MHz. It was shown (by comparing bit-by-bit all relevant signals) that performance and data integrity are maintained in all four cases.

As a visual demonstration, we send two closely separated test-pulses into channel 1, so that the rising edge of the second pulse makes its way into the first event. Fig. 12 shows scope traces of VME buffer contents for two clock speeds¹¹. Both events are successfully caught in the 150 MHz case, which was also verified

¹¹The output is iterated, so the third pulse in the 150 MHz case is simply a repetition of the first pulse

Figure 12: Two pulses caught at 100 and 150 MHz



doing a bit-by-bit comparison of digital data. Note that the jitterness in scope traces is caused by DAC and imprecise termination and is absent in the digital signal.

4 Conclusion

A trigger and data acquisition system was developed for the J-PARC experiment. Some preliminary tests were done using a Stratix II Development Board. These tests show promising results and suggest that it may be possible to implement the entire trigger and DAQ system (including VME readout logic) in an Altera FPGA device. We hope to use this design, with proper modifications, in a real-data test at Fermilab later this year.

5 Acknowledgments

I would like to thank Yau Wah, Harold Sanders, Mircea Bogdan, Fukun Tang, and Jim Pilcher for providing guidance in the course of this research.

References

- [1] Taku Yamanaka *et al.*, Proposal for the $K_L \rightarrow \pi^0 \nu \bar{\nu}$ Experiment at J-Parc, 2006
- [2] Meson Summary Tables (Review of Particle Physics, Particle Data Group), 2006
- [3] Jiansen Ma *et al.*, The Bessel Filter Simulation (internal note), 2007

- [4] Mircea Bogdan, JParc-K DAQ System (presentation at KEK), 2006
- [5] Altera Corp., Stratix II EP2S60 DSP Development Board Data Sheet, 2006
- [6] Altera Corp., Quartus II Development Software Handbook v6.1 (Complete Five-Volume Set), 2006
- [7] Altera Corp., Stratix II Device JTAG Configuration, 2004
- [8] Altera Corp., SignalTap II Embedded Logic Analyzer, 2005
- [9] Altera Corp., Implementing PLL reconfiguration in Stratix II Devices, 2004