

# Timing Studies of PULSAR & PC Communication for the Level 2 Trigger Upgrade

[K. Hahn](#), P. Keener, J. Kroll, C. Neu, H. F. Stabenau, D. Whiteson, P. Wittich  
R. Van Berg

*University of Pennsylvania*

## Abstract

This note presents measurements of the latency of S-LINK data transfer between a PULSAR & PC. Mock S-LINK data sources and drains were first used to provide latency and bandwidth estimates. Tests with two PC's allowed for preliminary round-trip timing measurements. Finally, we measured round-trip decision times using a PULSAR and PC in a scenario closely approximating the upgrade to the L2 trigger system. Our results show round-trip times near  $11 \mu\text{s}$  (including trigger algorithm execution), indicating that the PULSAR-PC design satisfies the specifications of the Level 2 trigger.

# Contents

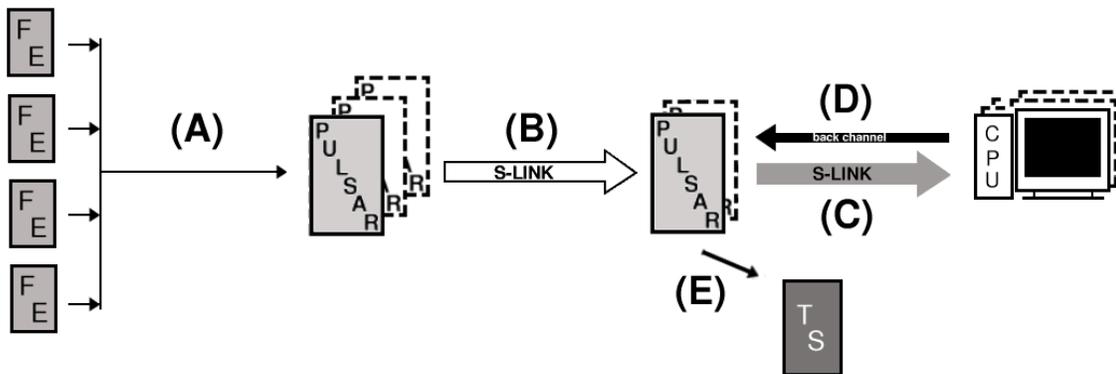
<b>1</b>	<b>Introduction</b>	<b>3</b>
<b>2</b>	<b>Hardware Overview</b>	<b>3</b>
<b>3</b>	<b>Test Details &amp; Results</b>	<b>5</b>
3.1	SLIDAS to PC timing using 126-word packets . . . . .	5
3.2	SLIDAS to PC to SLIDAD timings . . . . .	7
3.3	PC-PC testing . . . . .	9
3.4	PULSAR-PC testing . . . . .	10
<b>4</b>	<b>Conclusion</b>	<b>13</b>

# 1 Introduction

The goal of Level 2 upgrade project is to replace the existing Level 2 system with a commodity CPU fed by PULSAR boards [1, 2]. It is difficult to ensure that the extremely low latency requirements of the Level 2 trigger will be met by a design that uses commodity components. The Level 2 system is expected to accept data at rates of up to 40 kHz, which means that the total I/O loading and processing time, including the “back-channel” time to signal a decision, should not exceed approximately 20  $\mu$ s for dead-timeless operation. Conventional operating systems, as well as many networking devices, do not have latency guarantees at this level so it is necessary to carefully optimize both the algorithms and the accompanying I/O programming. This note documents studies that have been performed with the aim of establishing the necessary latency performance.

# 2 Hardware Overview

Figure 1 sketches the general configuration of hardware in the Level 2 trigger upgrade. The Level 2 trigger process begins as PULSAR boards receive Level 1 trigger information from the front-end detectors upon Level 1 accept (A). Additional PULSAR’s merge this data in an S-LINK stream (B) and direct it to the PC’s for evaluation (C). The PC’s perform the Level 2 algorithms and return their trigger decisions over a S-LINK “back-channel” to a PULSAR (D), which in turn informs the Trigger Supervisor of the result (E). The number of the PULSAR’s shown at each stage of Figure 1 is representative; the PULSAR design provides flexibility in the assignment of PULSAR’s to the segments of the Level 2 trigger process.



**Figure 1:** PULSAR’s in the Level 2 upgrade receive data from the front-end electronics. The data is formatted and routed to PC’s performing the L2 trigger algorithms.

In this note we detail timing measurements of the communication between the PULSAR and PC, labeled as arrows (C) and (D) in Figure 1. We first measured the

latency for S-LINK data transfer in one direction using a mock S-LINK data source and a PC. We next included a mock data drain that accepted data from the PC and measured the time for round-trip data transfer. By replacing the mock source and drain with a second PC that delivered event data and received trigger decisions, we were able to measure a round-trip time that accounts for the execution of the Level 2 trigger algorithms. Finally, we replaced the second PC with a PULSAR and again measured round-trip data transfer times that include algorithm processing. This test provides a reasonable estimation of the data transfer latencies of the real system as the only components used are those that will be present in the final design.

The Level 2 upgrade utilizes the optical fiber implementation of CERN’s S-LINK protocol [3] as a means of communication between input and output devices. These devices include CERN’s ODIN Link Source Card (LSC) and Link Destination Card (LDC) [4] cards that respectively serialize and de-serialize the data transferred on the optical link. The LSC and LDC cards may be attached to a variety of data sources and sinks. Our single PC tests use a LSC connected to a SLIDAS<sup>1</sup> [5] data source to introduce data on the link and an LDC attached to a S32PCI64 PCI card [6] to receive data on the computer. We also implement the reverse scenario, with the PC sourcing data through a S32PCI64/LSC combination to a SLIDAD<sup>2</sup> [7] and LDC. The SLIDAS/SLIDAD is connected to the LSC/LDC by means of the SLITEST PCB board. This device is used to manually send data to and receive data from the LSC/LDC cards. Both PC’s in the PC–PC tests use two S32PCI64 cards, one with a LSC and another with a LDC, enabling bi-directional data transfer between the two machines. In final tests, we replace the PC used to source event data (and to receive the trigger decision) with a PULSAR/auxiliary card pair [2]. The PULSAR directs data transfers to and from the PC through the LSC and LDC attached to the auxiliary card.

The S32PCI64 PCI card is used in all of our tests as the interface through which data on the S-Link is transferred to the computer. The device enables autonomous data reception on the PC through the use of direct memory access (DMA). Software on the PC initializes the S32PCI64 with addresses of PC memory locations, which the card uses to direct the DMA transfer of incoming data. Once initialized, the card is able to transfer data into main memory without support from software or the operating system. This operational independence allows the S32PCI64 to achieve throughput rates close to the limit of the PCI architecture [8].

The two PC’s used in our tests, labeled here as `pcpulsar` and `pcpulsar2`, run version 2.4 of the Linux kernel. `pcpulsar2` is the faster machine, equipped with two 2.4 GHz Intel Xeon processors and 1 GB of RAM. Two of its four independent PCI buses are dedicated to data input and output in our tests. `pcpulsar` has two 1.4 GHz Intel Pentium III processors and 1 GB of RAM. Its two independent PCI buses are shared between the I/O devices used in our tests and other system hardware. `pcpulsar` is used in the PC–PC tests described below to feed event data to `pcpulsar2`, and then to receive, check, and time the resulting trigger decisions. This leaves `pcpulsar2` to

---

<sup>1</sup>S-LINK Infinite Data Source

<sup>2</sup>S-LINK Infinite Data Drain

receive the event data, run the trigger algorithms and return decisions to pcpulsar. pcpulsar2 continues to perform these operations in PULSAR-PC tests, in which we use a PULSAR to replace pcpulsar as the data source/decision recipient.

The Pentium timestamp counter (TSC) is used to measure time intervals on the PC in most of our tests. We use the lower 32 bits of the TSC to achieve nanosecond resolution. A 32-bit counter on the PULSAR is used to measure time in PULSAR-PC tests. The counter derives from the 40 MHz S-LINK clock on board the PULSAR and has a resolution of 25 ns. In every test described, software that executes on the PC is run with real-time scheduling priority. Although we do not use a true real-time operating system that enforces strict time limits on system operation, the Linux kernel does contain a real-time scheduler that provides control over the extent to which processes share CPU resources. We adjust process priority through the scheduler via the `sched_setparam(2)` call to ensure that other system processes do not interfere with our tests.

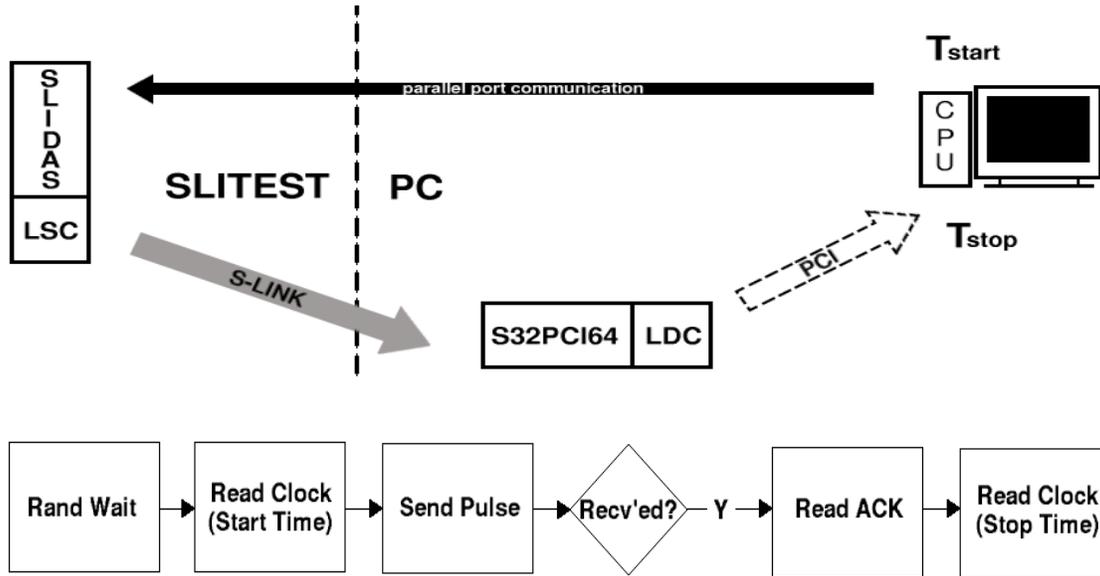
### 3 Test Details & Results

We tested individual components of the PULSAR/PC system in order to understand their latencies before testing the combined system. Measurements of the execution time of the trigger algorithms have been described in [9]. In this section we focus on tests of the I/O components, *i.e.*, tests of data loading time and back-channel time, as well as whole-system tests in which data is loaded and processed and the trigger decision is transmitted.

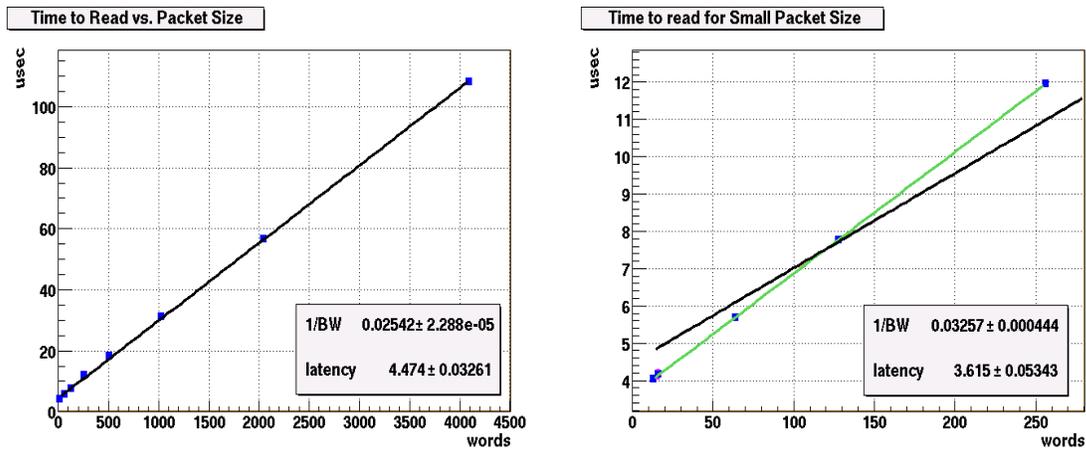
#### 3.1 SLIDAS to PC timing using 126-word packets

We first tested the baseline latency of our I/O components using a SLIDAS device connected to a S32PCI64/LDC pair on a Linux PC, as shown in the upper graphic of Figure 2. In this test, data generated on the SLIDAS is routed through the LSC and over a S-LINK connected to the PC. The PC initiates data transfers by toggling an output bit on its parallel port, which is tied to a trigger pin on the SLIDAS. As depicted in the lower graphic of Figure 2, test software first reads the TSC to establish a “start time” for data transfer before pulsing the parallel port. These operations are repeated until the state of the “ACK” PCI register on the PC indicates that a data transfer is complete. The PC then reads the TSC to determine a “stop time” for the transfer.

We used 126-word packets in the tests as this seemed a reasonable estimate of the mean packet size expected during high luminosity data taking. No attempt was made to eliminate the effect of interrupts in the CPU, which was found to be important in later tests, or to otherwise isolate the system from disturbances such as desktop usage and network activity. Using the parallel port to initiate timing measurements on the PC introduces extra overhead to this test that will not be present in the final system.



**Figure 2:** The hardware (top) and software (bottom) setups for the SLIDAS to PC test. The PC's parallel port is used to signal the SLIDAS and indicate the start of data transfer over the S-LINK. The PC checks a PCI register to determine the completion of a transfer.



**Figure 3:** Left: Total time vs. packet size for a range of packet sizes. The slope of the linear fit corresponds to inverse bandwidth and the intercept to the latency for data transfer. We operate in the small packet regime. Right: A closer view of total time vs. packet size for small packets. The smaller bandwidth fit describes small packet sizes best whereas the line of larger bandwidth indicates the best overall fit from the graph on the left.

It takes an unknown amount of time to pulse the parallel port on the PC to start data transfer and this time is included in our measurement. The results of the test are shown in Figure 3.

Our measurements indicate that the bandwidth for large packets is approximately 158 MB/sec (theoretical limit 160 MB/sec [4]) and the latency is approximately 4.5  $\mu$ s, which agrees with the hardware specifications from CERN. For small packets we find that the bandwidth is approximately 122.8 MB/sec, and the latency is approximately 3.6  $\mu$ s.

### 3.2 SLIDAS to PC to SLIDAD timings

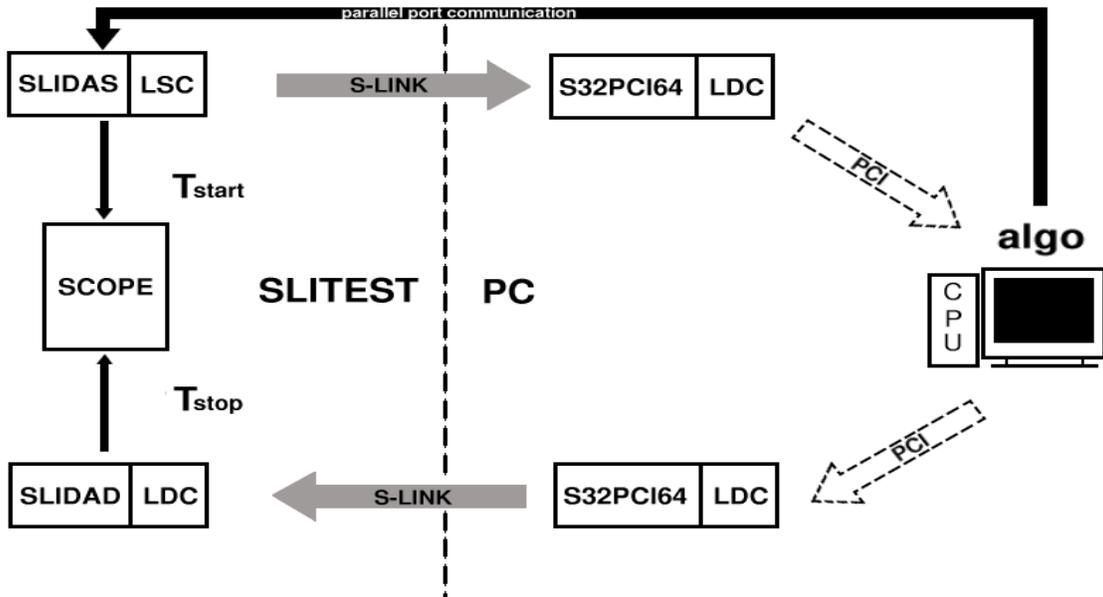
Next, we investigated the feasibility of using S-LINK as the back-channel for the L2 system. 126-word packets are sent from the SLIDAS and received on the PC, as in the ‘SLIDAS to PC’ test, but now the PC responds with a 100-word acknowledge packet sent to the SLIDAD (Figure 4). While a 100-word reply is much longer than the several word packet we expect to send over the back-channel, this length is used so that the reply is sufficiently long for it to be captured by the oscilloscope. Voltages corresponding to the 32 bits of each word generated by the SLIDAS and each word received by the SLIDAD are available on the boards’ diagnostic pins. A specific bit is chosen to distinguish the first word generated on the SLIDAS and the pin mapped to this bit is monitored by the oscilloscope to determine a start time for data transfer. Similarly, a specific bit is used to mark the last word generated on the SLIDAS and the oscilloscope measures a “stop time” when this bit appears on the diagnostic pins of the SLIDAD. The PC initiates data transfers by triggering the SLIDAS via the parallel port, as is done in the SLIDAS-PC test. In this test, however, the time needed to trigger the SLIDAS is not included in the measurement since the diagnostic pins provide an independent means of determining when data transfers begin and complete.

At this point we wanted to determine the effects of hardware interrupts on our measurement. Interrupts occur when hardware in the PC requests the CPU’s attention, interrupting sequential program execution until the CPU finishes servicing the request. In order to eliminate interference from interrupts in our tests, our code was written as a kernel module.<sup>3</sup> Measurements of round-trip time from the SLIDAS-PC-SLIDAD tests taken with and without interrupts enabled are shown in Figure 5.

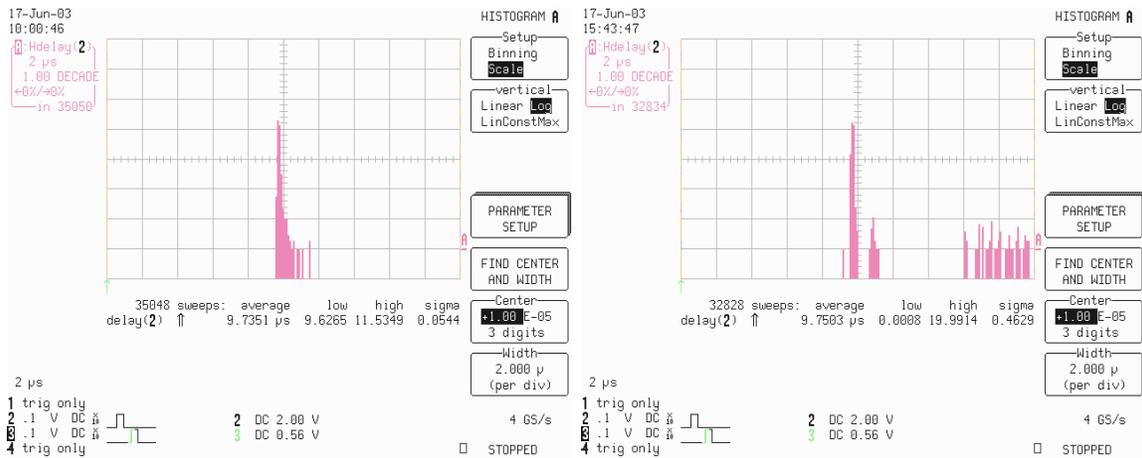
The histogram of round-trip time taken with interrupts enabled, displayed on the right in Figure 5, shows a small number data transfers that take much more time than others to complete. Because the leftmost histogram of Figure 5 (taken with interrupts disabled) does not contain such events, we conclude that these tail events result from interrupts. Based on these measurements, we see that we can achieve a total I/O loading time that is on average about 10  $\mu$ s and no slower than 12  $\mu$ s by using a combination of real-time scheduling and interrupt binding on a commodity Linux-PC system with S-LINK “front” and “back” channels. It seems then that this

---

<sup>3</sup>It was later discovered that it is possible on a Linux-SMP machine to have one CPU handle all peripheral interrupts, achieving the same effect in a much simpler way.



**Figure 4:** A schematic of the SLIDAS-PC-SLIDAD test. An oscilloscope is used to measure the time between the start of data transfer on the SLIDAS and the receipt of the PC's reply on the SLIDAD.

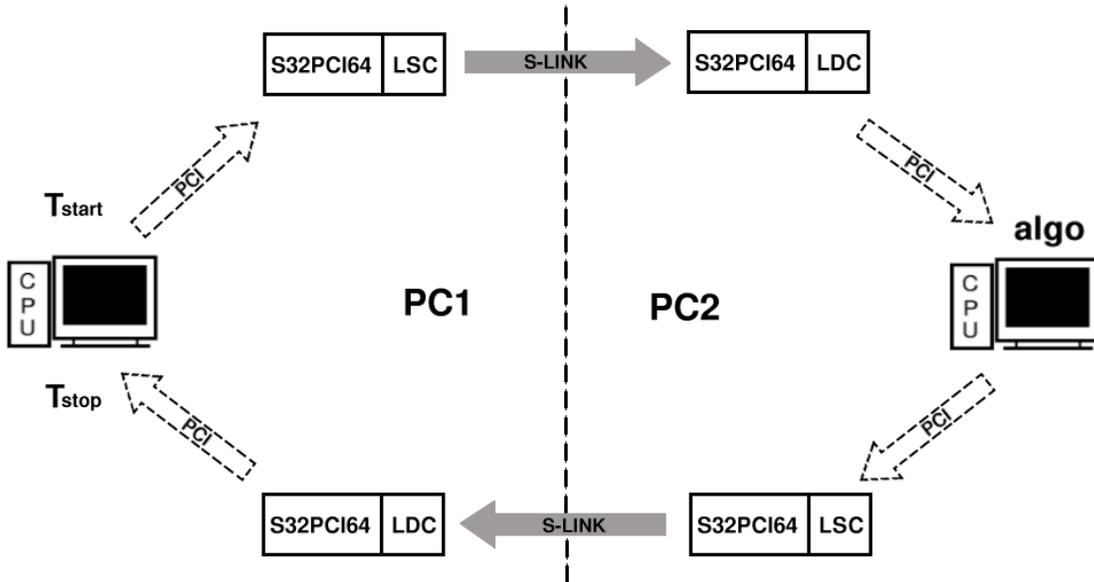


**Figure 5:** Left: Test results with interrupts disabled. The longest event takes approximately 11  $\mu$ s. Right: Test results with interrupts enabled. The tail of the distribution exceeds the measurement range at 20  $\mu$ s. This behavior results from hardware interrupts that interfere with the execution of our test software.

hardware/software combination is a good candidate for the transport layer of this system.

### 3.3 PC-PC testing

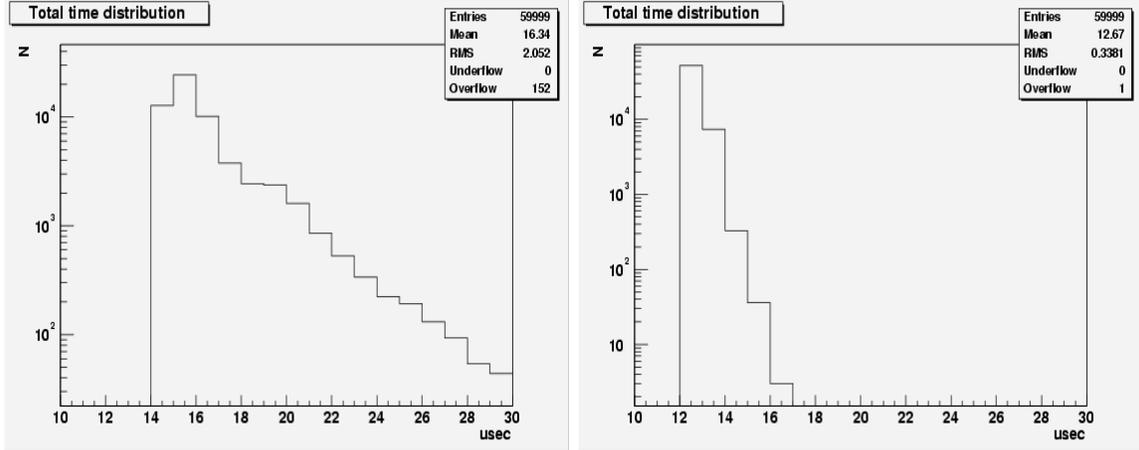
Having accomplished a complete test of the I/O subsystem, we desired to create as realistic a test as possible with two PC's. We attached a second PC to the algorithm PC, which sends event data, receives and checks the decision bitmask, and records the round-trip time. The hardware arrangement for this test is depicted in Figure 6. Recall that we implement code running on the PC in the SLIDAS-PC-SLIDAD tests as a kernel module to avoid interference from peripheral interrupts. We achieve the same effect in this test using a far simpler approach. The Linux operating system permits users to bind hardware interrupts to a particular CPU (or CPU's) of a multi-processor machine. This feature of the operating system, named "smp-affinity" [10], allows us to stipulate the handling of peripheral interrupts by one CPU, while the I/O-algorithm code runs with full scheduling priority on the other.



**Figure 6:** PC-PC test schematic. Timing starts when PC1 sends event data over a S-LINK to PC2, which uses the data as input to the trigger algorithms. Timing stops once PC1 receives a trigger decision sent over another S-LINK from PC2.

The results of the PC-PC testing are provided in Figure 7 and give a baseline for the worst-case performance we expect from our I/O system. We expect better performance from the real system since the use of a second PC instead of a PULSAR in the test forces data to be transferred over more PCI buses, resulting in increased latency that will not be present in the final configuration. Note that as in the previous test, contention for the PCI bus between the cards that send and receive data is not an issue since the machine has two independent PCI buses. Having eliminated the effect

of interrupts, we see that our mock trigger setup is I/O bound in the majority of cases, but turns CPU-bound when the algorithms take a long time.



**Figure 7:** Left: Round-trip time for PC-PC testing with the trigger algorithms performed. Right: Round-trip time for PC-PC testing without the trigger algorithms performed.

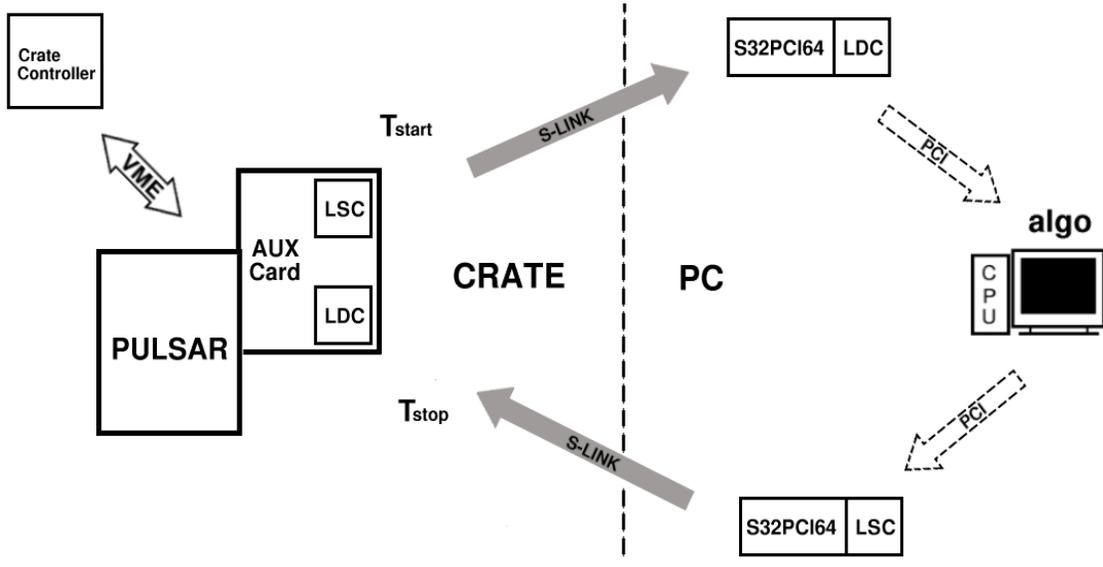
### 3.4 PULSAR-PC testing

Finally, we performed tests with a PULSAR sourcing data to and receiving trigger decisions from the PC. In our tests a PULSAR sends and receives data by means of a 9U S-LINK auxiliary card supporting a CERN S-LINK LSC and LDC, as shown in Figure 8. Firmware<sup>4</sup> and software were developed to allow data transfers to be driven by the VME controller that shares a CDF crate with the PULSAR. The controller first writes data for an event to a RAM on the PULSAR before instructing the PULSAR to send the contents of the RAM over the S-LINK to the PC. The PC (pcpulsar2) receives the event data on its LSC, runs the trigger algorithms and generates a decision bitmask, which is sent over the “back-channel” via its LSC. The bitmask is received on the PULSAR’s LDC and latched into another RAM on the PULSAR. Once latched, decision bitmasks may be read out via VME for comparison with the expected decision mask from the TL2D bank.

Round-trip timing was performed on the PULSAR using a 32-bit counter provided in firmware. The counter starts on the beginning of data transfer and stops when the last word of the decision mask is latched into the receiving buffer. The time difference (*i.e.*, the round-trip time) is read out using VME instructions before beginning data transfer for the next event.

The hardware configuration used in our tests very closely approximates the system design discussed in the PULSAR proposal [1]. The real system will also include a

<sup>4</sup>The firmware for our tests was provided by Sakari Pitkanen and Tomi Mansikkala.



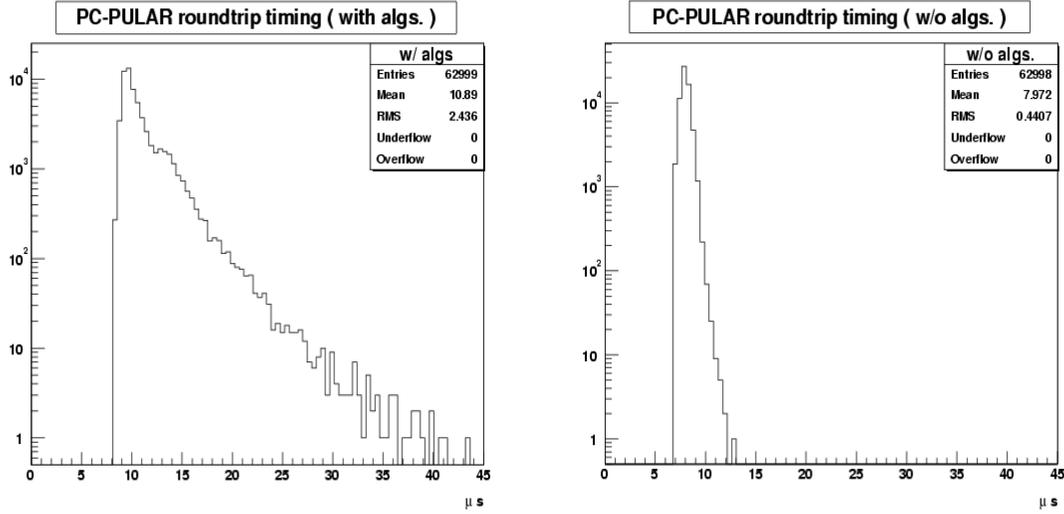
**Figure 8:** PULSAR-PC test schematic. The PULSAR loads event data from a crate controller and starts its on-board timer as the S-LINK transfer to the PC begins. The PC performs the algorithms and returns a trigger decision over S-LINK to the PULSAR, which then stops its on-board timer.

Trigger Supervisor (TS), which receives trigger decisions from a PULSAR (see label E of Figure 1). Our tests do not account for the time necessary to relay trigger decisions to a TS.<sup>5</sup> Additionally, the final system may include two different PULSAR’s that communicate with the PC, one to input data and another to receive the trigger decision. Our tests employ a single PULSAR to accomplish both tasks, however we do not expect that the use of separate PULSAR’s will result in round-trip times different from those we present.

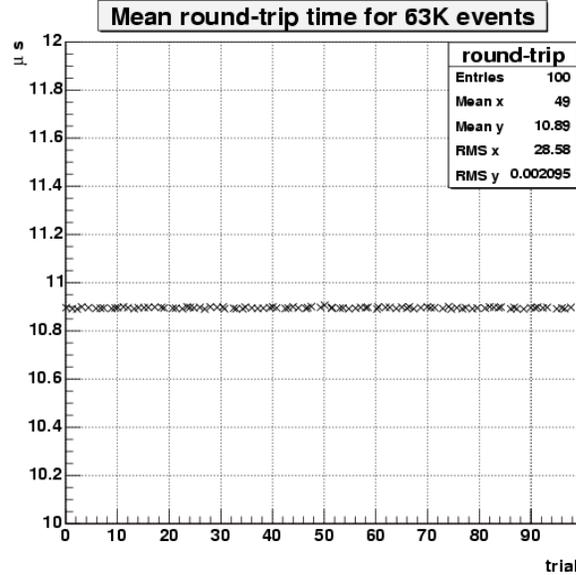
The plots in Figure 9 show the round-trip times of 63K events taken with and without the trigger algorithms running. The PC sends an empty four word bitmask back to the PULSAR in tests where the algorithms are not run. The data we used was stripped from a PHYSICS\_1.05-v8 dataset to form a 80%-20% mixture of track/no-track events. We zero-suppress and format the stripped data as per the proposed PULSAR/S-LINK data format [11]. The mean event size for the reformatted dataset is 34 words. We verify the trigger decisions for all events in the distributions shown against those obtained from the TL2D bank.

The round-trip times shown in Figure 9 are smaller than those seen in the PC-PC distributions of Section 3.3. The difference likely stems from the additional PCI bus transfers of the second PC, now absent in this test, and the smaller (more realistic) data size. The difference between the means of the algorithm and no-algorithm distributions

<sup>5</sup>The communication of a trigger decision from the Alpha processor to the TS currently requires  $1.7 \mu\text{s}$  on average.

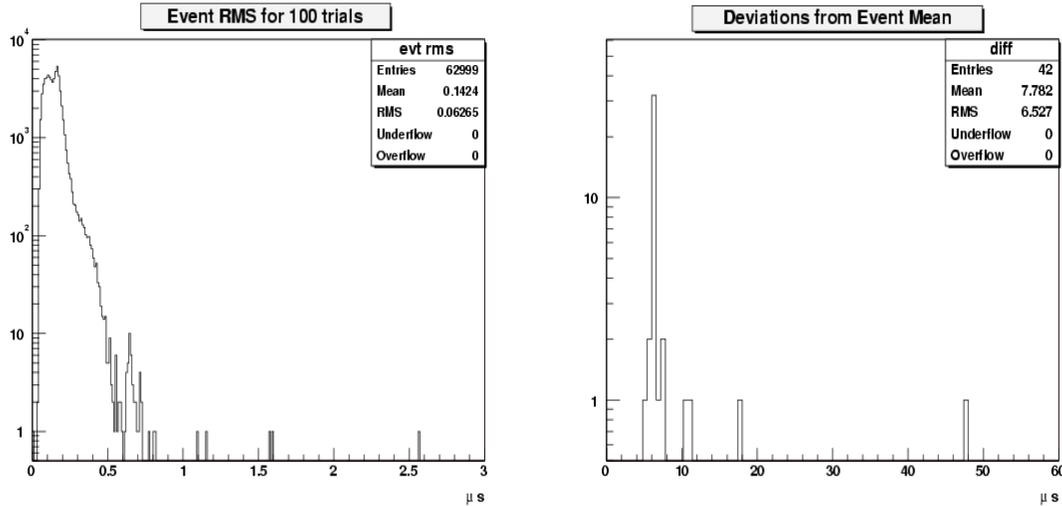


**Figure 9:** Left: Round-trip time for PULSAR-PC testing with algorithms performed. Right: Round-trip time for PULSAR-PC testing without algorithms performed.



**Figure 10:** Mean round-trip time (of 63K events with algorithms) for 100 trials.

is consistent with earlier stand-alone measurements of algorithm time and that the number of tail events appearing in the algorithm distribution is also consistent with those measurements [9]. The plots provided in Figure 9 represent one of 100 trials performed with the 63K event dataset. The mean round-trip times from each of the



**Figure 11:** Left: RMS’s of the round-trip time for 100 trials of 63K events. Right: Events with round-trip times that deviate significantly from their mean value occur in our tests at the level of 7 ppm. This plot shows the difference between the mean and measured round-trip time for these outlying events.

100 trials (with algorithms) are provided in Figure 10 and demonstrate that the value of the mean shown above is stable.

The leftmost histogram of Figure 11 provides the RMS of round-trip time for every event in the 63K event dataset. Each event was repeated 100 times, corresponding to the 100 trials mentioned previously. The plot shows a surprisingly small spread in round-trip time, considering that the operating system we use is not “real-time”. Although the RMS on round-trip time is small, events in several trials show round-trip times that deviate significantly from their mean values. The rightmost histogram of Figure 11 displays the difference between the mean and measured round-trip time for events in which this difference exceeds  $5\mu\text{s}$ . Since the plot contains entries for 42 events, we conclude that such extreme outliers occur at a rate of about 7 ppm. As in PC-PC testing, all peripheral interrupts were blocked from reaching the CPU executing the trigger algorithms and the I/O-algorithm code was scheduled to run at maximal priority.

## 4 Conclusion

We have shown that it is possible to achieve I/O latencies of around  $11\mu\text{s}$  for a Level 2 trigger consisting of a PULSAR board and a commodity dual-processor PC that communicate via CERN’s optical S-LINK. The small RMS’s on the round-trip times measured in our tests indicate stable operation in this regime. The latency value

was obtained using real event data and includes the execution time of the Level 2 trigger algorithms. The system performance we have demonstrated is achieved using a standard Linux distribution, without recourse to a real-time operating system. Given the results, our tests constitute a successful “proof of principle” for the PULSAR/PC design.

## References

- [1] CDF note CDF/DOC/TRIGGER/CDFR/6259
- [2] <http://hep.uchicago.edu/~thliu/projects/Pulsar/>
- [3] <http://hsi.web.cern.ch/HSI/s-link/>
- [4] <http://hsi.web.cern.ch/HSI/s-link/devices/odin/>
- [5] <http://hsi.web.cern.ch/HSI/s-link/devices/slidas/>
- [6] <http://hsi.web.cern.ch/HSI/s-link/devices/s32pci64/>
- [7] <http://hsi.web.cern.ch/HSI/s-link/devices/slidad/>
- [8] [http://edms.cern.ch/cedar/plsql/doc.info?document\\_id=315809](http://edms.cern.ch/cedar/plsql/doc.info?document_id=315809)
- [9] CDF note CDF/DOC/TRIGGER/CDFR/6620
- [10] <http://www.kernel.org/pub/linux/kernel/people/rml/cpu-affinity/>
- [11] <http://fozzie.uchicago.edu/~thliu/Pulsar/meetings/030425/CJL.pdf>