exploratory "by-product". The dream part: to have a clear theoretical connection between such observations and the existence of the critical point in the QCD phase diagram.

In order to reach these goals, we would need to put together an improved NA60-like experiment, which could operate in the period 2010–2014. It is important to keep in mind that, despite the very important guidance from theory, significant progress in the field crucially depends on high accuracy measurements. An accurate measurement at the SPS can be more useful to push progress than a "lousy" measurement at higher (or lower) energies. The new NA60-like experiment should collect data in the following conditions.

- ▷ Pb-Pb at 158 GeV/nucleon: improved study of charmonia with respect to the NA50 data and a completely new study of low and intermediate mass dimuons at the highest SPS energy densities.
- ▷ Pb-Pb at 50 GeV/nucleon: measure the ρ spectral function at higher baryon densities and lower temperatures; ideally it should be complemented by shorter runs (of about one week) with other A-A systems and/or energies.
- ▷ Cu-Cu at 158 GeV/nucleon: to establish the "normal J/ψ nuclear absorption" before new physics sets in and to confirm the location of the critical energy density (ϵ is not well defined in p-A collisions).
- ▷ Pb-Be at 158 GeV/nucleon: "inverse kinematics" experiment, to measure nuclear effects in charmonia production at negative x_F (down to around -0.7).

If and when the SPS will be upgraded in view of the LHC luminosity upgrade, the so-called SPS+ scenario, with two times higher energies, such a new experiment could also take Pb-Pb data at higher energies. In particular, this would be very interesting to confirm, with high statistics, that the J/ψ survival probability flattens out at 0.6, until it is further suppressed at the LHC due to the dissociation of the directly produced J/ψ mesons (if thermal regeneration of charmonia states does not take over).

Beyond the future

The only way to discover the limits of the possible
is to go beyond them into the impossible.
Arthur C. Clarke

Speaking of LHC, I cannot help expressing my personal dream of one day using the 7 TeV proton beam in a fixed-target setup, which coupled to a high-luminosity slow extraction could provide very interesting data on rare or forbidden decays in the Standard Model which involve muons, like the $D^0 \to \mu\mu$ decay. Setting limits on the branching fractions of these decays restricts the parameter space of many proposed extensions of the Standard Model, while an actual observation would be a manifestation of physics beyond the Standard Model.

Appended matter

In this part

Appendix A

NA60's Silicon Pixel Tracker for indium running

This appendix collects two publications related to the NA60 silicon pixel tracker in view of, or as used in, the indium run of 2003. These publications cover hardware, operation and performance aspects. For that reason, it is a very important complement to both chapters 2 and 6.

Section A.1 details work prior to the 2003 indium run, namely individual assembly performance tests and a test-run with three small pixel planes in real experimental conditions which took place in October 2002 with a low-energy lead ion beam.

Section A.2 describes the detector's performance during the indium run in 2003 with emphasis on the accumulated radiation damage and vertexing abilities.

Available online at www.sciencedirect.com
SCIENCE DIRECT®
ELSEVIER
Nuclear Instruments and Methods in Physics Research A 539 (2005) 137–145
NUCLEAR INSTRUMENTS & METHODS IN PHYSICS RESEARCH Section A
www.elsevier.com/locate/nima

Operation and performance of the NA60 silicon pixel telescope

K. Banicz[a,e], A. David[a,b], M. Floris[c], J.M. Heuser[d], M. Keil[a,*], C. Lourenço[a], H. Ohnishi[d], E. Radermacher[a], R. Shahoyan[b], G. Usai[c]

[a]*PH Department, CERN, 1211 Geneva 23, Switzerland*
[b]*Instituto Superior Técnico, Lisbon, Portugal*
[c]*Università di Cagliari and INFN, Cagliari, Italy*
[d]*RIKEN - The Institute of Physical and Chemical Research, Wako, Saitama, Japan*
[e]*Now at Universität Heidelberg, Heidelberg, Germany*

Received 19 May 2004; received in revised form 7 September 2004; accepted 8 October 2004
Available online 19 November 2004

Abstract

The NA60 experiment studies open charm and prompt dimuon production in proton–nucleus and nucleus–nucleus collisions at the CERN SPS. The high multiplicity of charged tracks produced in heavy-ion collisions imposes the use of silicon pixel detectors to perform an efficient tracking. This paper describes the design and assembly of the pixel telescope and performance results from three detector planes operated in the high charged particle multiplicity conditions of Pb–Pb collisions.

PACS: 29.40.Gx; 29.40.Wk

Keywords: Pixel detector; Vertex detector

1. Introduction

The NA60 experiment [1] aims at studying the production of prompt dimuons and open charm in proton–nucleus and heavy-ion collisions at the CERN SPS. New state-of-the-art silicon detectors in the vertex region complement the muon spectrometer and zero degree calorimeter previously used in NA50. A radiation tolerant beam tracker, based on silicon microstrip detectors operated at 130 K, measures the transverse coordinates of the incident ions with a precision of approximately 20 μm. Together with the information from the vertex pixel telescope this allows us to measure the offset of the muon tracks with respect to the event vertex and thereby distinguish events where a pair of D mesons was produced in the collision. So far, the NA60 data taking periods

*Corresponding author. Tel.: +41 22 767 9780; fax: +41 22 767 9487.
E-mail address: markus.keil@cern.ch (M. Keil).

0168-9002/$ - see front matter © 2004 Elsevier B.V. All rights reserved.
doi:10.1016/j.nima.2004.10.019

include proton–nucleus and low energy Pb–Pb collisions in 2002 and high energy In–In collisions in 2003.

2. The NA60 silicon pixel telescope

Downstream of the target and inside a 2.5 T magnetic dipole field, a silicon tracking telescope measures the charged tracks and allows us to identify which of them provide the best match with the muons measured in the muon spectrometer, which is placed behind a hadron absorber. For proton runs, this tracking telescope can consist of silicon microstrip planes, whereas the high particle multiplicity in heavy ion collisions imposes the use of silicon pixel detectors.

The pixel telescope which NA60 developed for the 2003 Indium run consists of 16 independent detector planes with 96 pixel chip assemblies in total, arranged along the beam axis over a length of ~26 cm, starting at ~7 cm downstream of the centre of the target. To optimally subtend the angular acceptance of the muon spectrometer, $3<\eta<4$ or $35<\theta<120$ mrad, the first planes are smaller than the last ones. Fig. 1 shows the arrangement of the pixel detector planes. Each plane consists of several single-chip pixel detector assemblies which are mounted on a planar ceramic support ('hybrid'). The assemblies are made from radiation tolerant, 750 μm thick ALICE1LHCb pixel readout chips [2], bump bonded to 300 μm thick p-on-n silicon pixel sensors with $425\times 50\,\mu m^2$ pixels. The detector planes closest to the target comprise 4 of such assemblies, while the planes further downstream comprise 8 assemblies per ceramic, with two ceramics mounted back-to-back to form one logical plane (Fig. 2). Since the pixel cells are not square, they measure one of the coordinates of the hit, x or y, with a better precision, depending on their orientation when positioned on the planes. To optimise the tracking and vertexing performance of the pixel telescope, the small planes exist in 'x' and 'y' versions, while the large ones are all 'x', having the best resolution coordinate perpendicular to the magnetic field lines, to optimise the momentum measurement. Some planes are placed with

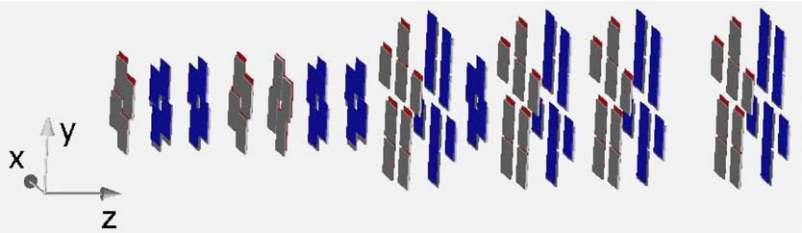


Fig. 1. The complete vertex telescope, consisting of 4-chip and 8-chip planes. The beam comes from the left, the magnetic field is parallel to the y-direction.

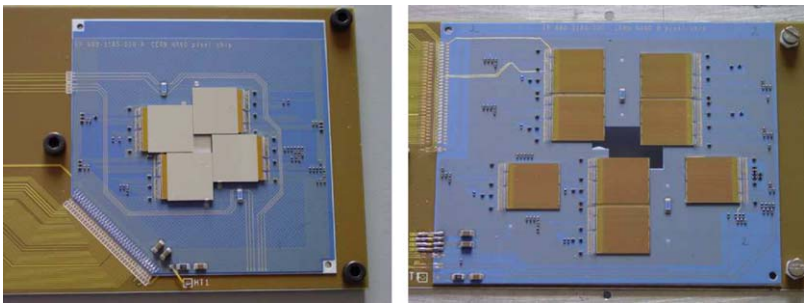


Fig. 2. A 4-chip and an 8-chip hybrid.

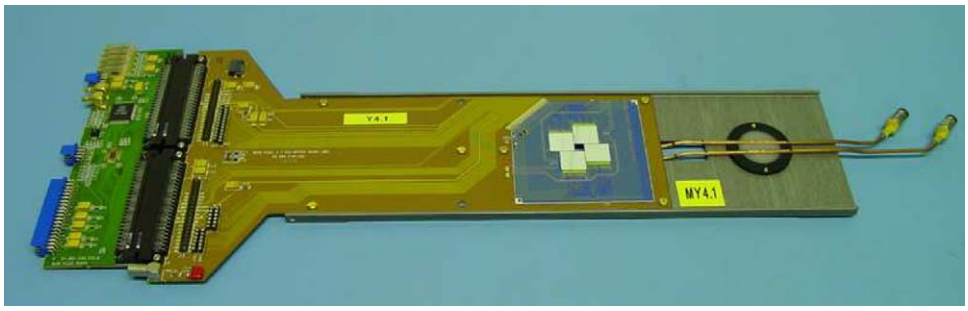


Fig. 3. A 4-chip pixel detector plane with aluminium frame and cooling tubes.

inverted orientation to maximise the overall angular acceptance coverage.

Each of the single chip assemblies consists of a matrix of 32 columns and 256 rows, resulting in a total of 8192 pixels. In the readout chip, every pixel cell contains a preamplifier, a shaper and a discriminator, as well as control and readout logic to transfer the binary hit information to the wire bond pads at the edge of the chip [2]. The collected and amplified charge is digitised by the discriminator whose threshold can be set by a global 8-bit voltage DAC and individually adjusted for each pixel cell by a local 3-bit DAC. Four "test columns" have additional diagnostic outputs at different stages of one readout cell. The chip is operated at 10 MHz and read out over 32 parallel data lines. The chips on one plane are read out sequentially. They were produced in a 0.25 μm technology and have been shown to remain fully functional after exposure to radiation doses of at least 12 Mrad [3]. Therefore, the detector should cope with the radiation levels of ~ 1 Mrad per week expected in the cells closest to the beam axis on the first planes of the NA60 pixel telescope.

The readout chips are glued to the multilayer hybrid circuit fabricated on Al_2O_3 or BeO substrates. The ceramic support itself is glued and wire bonded onto a printed circuit board that provides mechanical support and routes all signals between the front-end electronics and the data acquisition, control and power supply systems. The circuit boards, mounted on aluminium frames, fit into slots of a support box between the dipole magnet shims, defining the detector positions with respect to the targets. The modules are cooled by water at ~12°C circulating in a copper tube attached to the back of the hybrid. Fig. 3 shows a fully mounted 4-chip plane with the aluminium frame and cooling tubes.

3. Tests of assemblies and modules

Prior to the installation in the experiment, comprehensive quality assurance and system tests were performed in the laboratory. Electrical tests and measurements with radioactive sources yielded threshold behaviour, the number of dead pixels and pixels with missing bump connections of the individual assemblies. Assemblies with more than 96% of pixels working were used for the construction of the planes. The assembled planes were then tested and calibrated electrically for the operation in the experiment, using the internal pulser of the readout chip. Maps of dead and noisy pixels were prepared using a ^{90}Sr source.

A common threshold voltage for all pixels in the ALICE1LHCb chip is set by an internal DAC called pre_VTH. The actual threshold value for a given pre_VTH setting is measured by injection of test charges with a varying pulse height. Measuring the response of single pixels as a function of the pulse height gives integrated gaussian distributions whose 50%-point defines the pixel threshold, whereas its width corresponds to the electronics noise. Fig. 4 shows how the measured average threshold depends on the value set on the pre_VTH DAC, for each pixel chip assembly of a given plane. One can see a linear behaviour of the measured average threshold with the pre-set

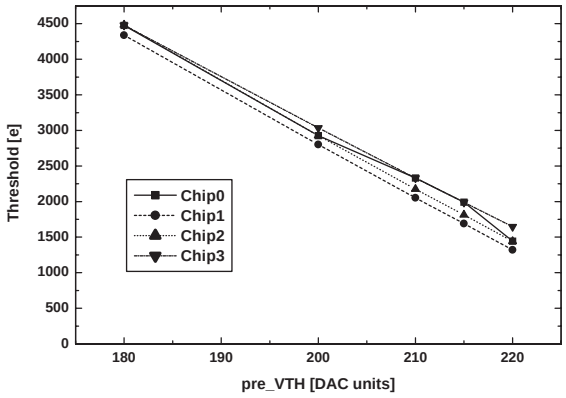


Fig. 4. Threshold calibration for all chips of a pixel detector plane. The measured average thresholds show a linear dependence on the setting of the global threshold DAC. The value of this DAC can be adjusted individually for every chip.

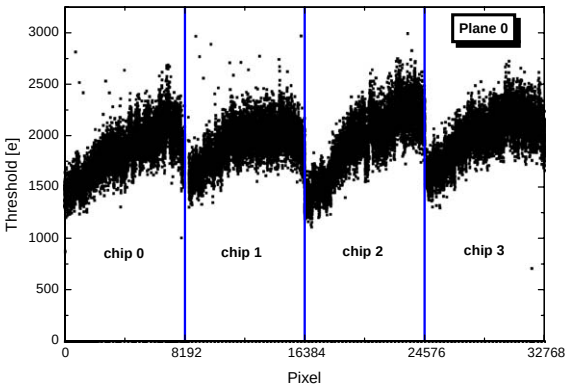


Fig. 5. Measured thresholds for all pixels of a 4-chip plane, for a nominal threshold of 2000 e and without pixel-to-pixel threshold adjustment.

value of the pre_VTH DAC. Thanks to low noise values and small pixel-to-pixel variations of the threshold, a nominal threshold value of 2000 e could be used in the experiment. The measured thresholds for a whole plane at this threshold setting can be seen in Fig. 5. The measurement shown yielded a threshold distribution with a mean value of 1930 e and a width of 260 e. Given the level of signal (approximately 30,000 e average) and of noise ($\sim$ 200 e) a dispersion of 200–300 e on a threshold set at 2000 e does not affect in any visible way the performance of the detector. Therefore, there was no need to use the pixel-to-pixel adjustment, a complex fine tuning procedure, which would have required some effort and a thorough understanding of the sources of the measured threshold dispersion.

4. Results from the 2002 lead ion run

Three 4-chip pixel detector planes were operated in the NA60 experiment during the data taking period of October 2002, when we studied Pb–Pb collisions, at 30 and at 20 GeV per incident nucleon, 5 days at each energy. The target system in this run consisted of three Pb targets of 0.5, 1.0 and 1.5 mm thickness. The pixel detector planes were positioned at 7.8, 10.1 and 13.1 cm downstream from the centre of the target box. Data samples have been collected with two types of triggers, both provided by the zero degree calorimeter of NA60: the "minimum bias" trigger simply required a minimal energy deposited in the calorimeter, to stay above the intrinsic noise of the detector; the "interaction" trigger had the extra condition that this energy was significantly below the total energy of the incident Pb ion, to ensure that an interaction had occurred in the target and the ZDC only measured the energy of the non-interacting beam nucleons. No magnetic dipole field was applied during the data taking.

Fig. 6 shows the number of reconstructed tracks for minimum bias Pb–Pb collisions. A relatively high track density of up to 100 or more tracks per event can be seen, leading to an occupancy of up to 2×10^{-2} hits per pixel per trigger in the pixels close to the beam hole of the detector planes. Fig. 7 shows a typical hit map in one plane, integrated over several minutes.

4.1. Cluster sizes

The behaviour of the cluster sizes has been studied in detail for all three planes since it is directly related to the detector occupancy and could be a limiting factor in forthcoming heavy ion runs.

The collected data sample shows that the cluster size mainly depends on three factors:

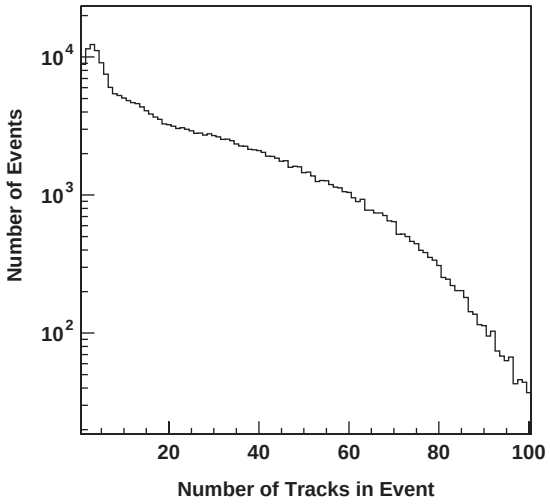


Fig. 6. Distribution of the number of tracks per event reconstructed in the vertex telescope.

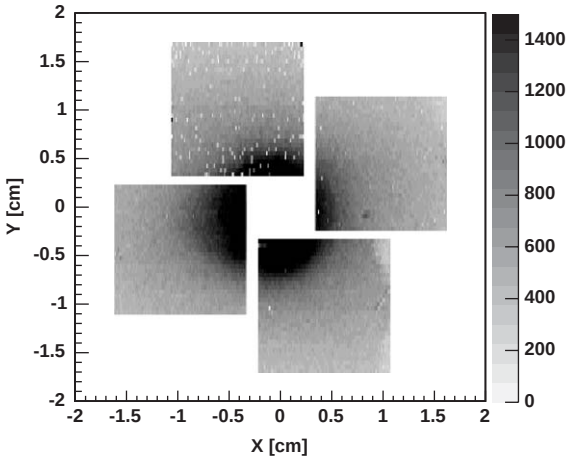


Fig. 7. Hit map of a pixel detector plane integrated over several minutes of Pb–Pb running.

Track angle—Fig. 8 shows the average cluster size versus the slope of the track with respect to the short pixel direction. To exclude the effect of bigger clusters due to delta electrons, clusters of more than three pixels were excluded. The plot shows the behaviour expected from geometrical considerations: with increasing track inclination the number of neighbouring pixels crossed by the particle, and thus the cluster size, increases.

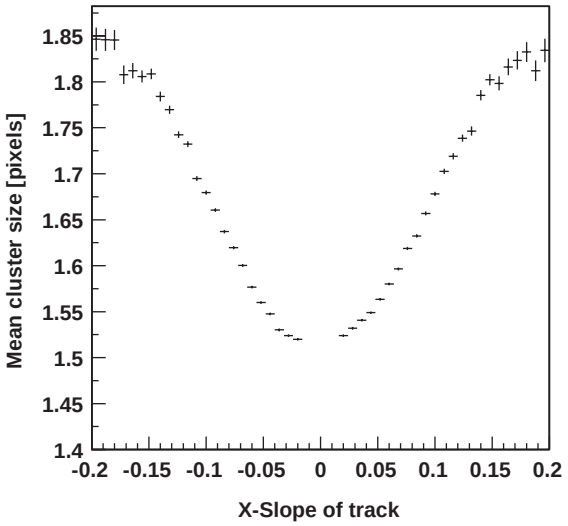


Fig. 8. Dependence of the mean cluster size on the slope of the track, dx/dz, with respect to the short pixel direction. For inclined tracks the charge tends to be deposited in several pixels.

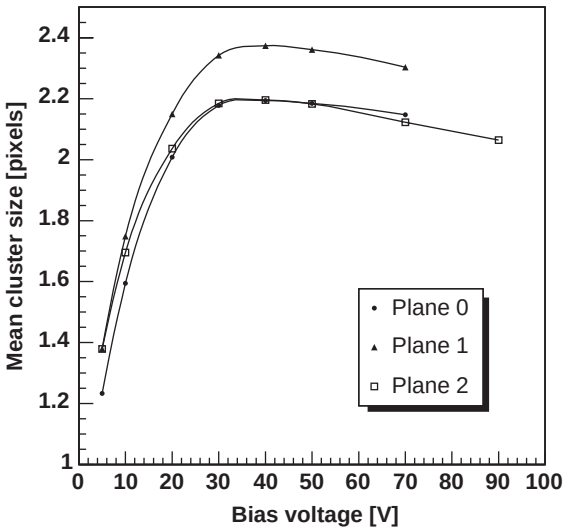


Fig. 9. Dependence of the mean cluster size on the detector bias voltage, for the three planes.

Charge collection—As can be seen from Fig. 9, the average cluster size increases with increasing bias voltage, until the detector reaches the expected point of full depletion (between 30 and 40 V). A further increase of the bias voltage leads

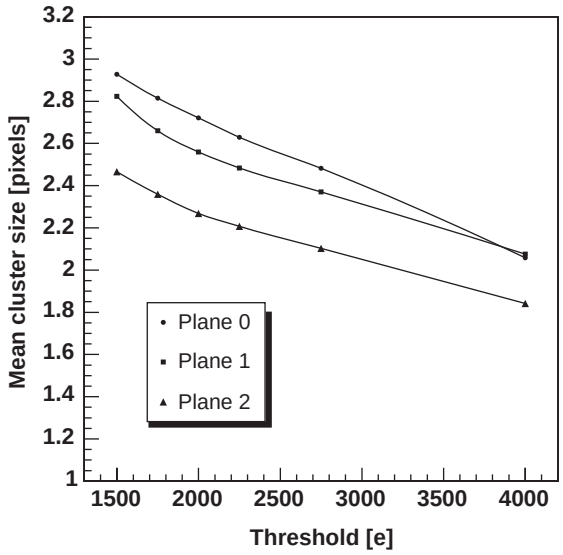


Fig. 10. Dependence of the mean cluster size on the nominal discriminator threshold for all three planes.

to a faster charge collection and to a smaller charge spread parallel to the sensor surface, resulting in a decrease of the cluster size.

Applied threshold—Fig. 10 shows the dependence of the cluster size on the setting of the discriminator threshold. As expected, the cluster size decreases with increasing threshold since, especially in big clusters, many pixels collect only small fractions of the signal charge.

The behaviour of the cluster sizes can, to a good extent, be understood by the effects described above. The threshold setting can be used to lower the occupancy of the pixel planes when necessary, provided the efficiency does not change when increasing the threshold (discussed below).

4.2. Track reconstruction

The spatial resolution and the efficiency of the pixel detector were assessed as follows. In a first step all tracks in the vertex telescope were reconstructed for a given event, requiring that each track has hits in all three planes. After identifying all vertices inside the target region, the vertex with the largest number of attached tracks was chosen. Then a second track reconstruction was done, this time requiring hits in all planes except the one whose efficiency and resolution was to be determined. Furthermore, an attachment to the vertex found in the first step was required, to reduce the combinatorial background. The tracks found in this step were extrapolated onto the plane under study, and that plane was checked for the presence of a hit in the neighbourhood of the extrapolation point. If no cluster could be found within 3σ of the track extrapolation, the chip was considered inefficient for this track.

4.2.1. Efficiency

Fig. 11 shows the efficiencies determined for a single chip versus the y position, corresponding to the chip columns, of the extrapolated track. The efficiency is almost 100% except for the regions of the so-called "test columns", whose efficiency was compromised by non-optimised DAC settings. Excluding the 4 points obviously affected by these columns, we get an average efficiency of 99.0%. No dead or noisy pixels have been excluded from this calculation.

Fig. 12 shows the average efficiency of a complete chip, including the test columns, measured during a threshold scan. We can see that, for thresholds between 1500 e and 4000 e, the efficiency stays constant within the precision of the measurement. This was expected since the average charge of approximately 30,000 e deposited in the sensor is much larger than the thresholds set.

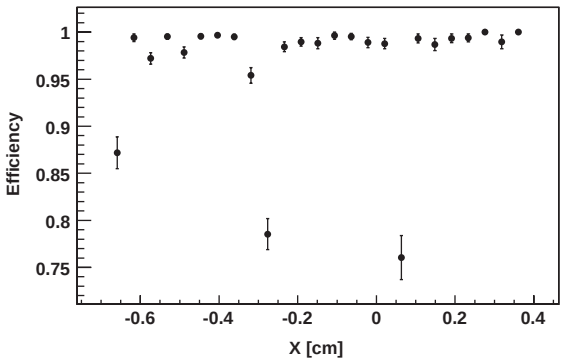


Fig. 11. Efficiency measured for one chip of plane 1. The regions of lower efficiencies are due to the existence of "test columns". Their efficiencies can be much higher when using fine tuned chip settings, not done when the data was taken.

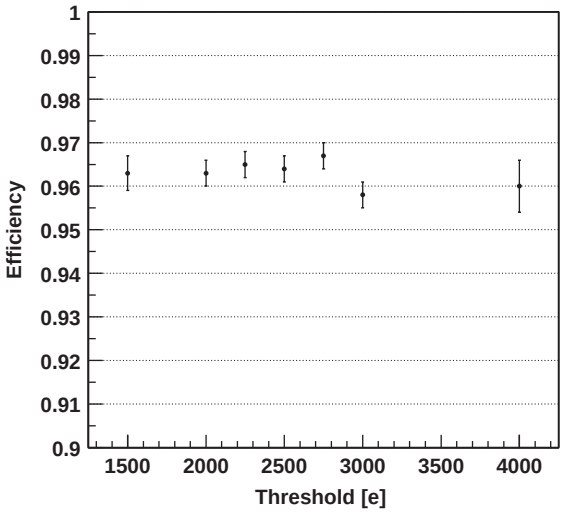


Fig. 12. Dependence of the measured efficiency on the setting of the discriminator threshold. The efficiency has been averaged over the complete chip, including the test columns.

4.2.2. Spatial resolution

Comparing the positions of the extrapolated track points with the hits found in the plane under study yields residual distributions, from which we can derive the spatial resolution. An example of such a distribution is shown in Fig. 13 for the 50 μm pitch direction. The distribution can be described by a convolution of the intrinsic resolution of the pixel plane under study with the resolution function of the track extrapolation. The width of the residual distribution is approximately 13 μm for both 1-pixel and 2-pixel clusters. The contribution of the track resolution was estimated assuming that for a given cluster pattern the spatial resolution is equal in all three planes. Under this assumption we could deconvolute the two contributions, obtaining an intrinsic spatial resolution of approximately 10 μm for both 1- and 2-pixel clusters. At the same time equal numbers of 1- and 2-pixel clusters were observed in the data. This indicates that approximately half of the pixel width contributes to each class, since the ratio between 1-pixel clusters and 2-pixel clusters is directly related to the fraction of the pixel area in which tracks cause single hits or double hits, respectively. This is in agreement with the spatial resolutions measured for both cluster types. The

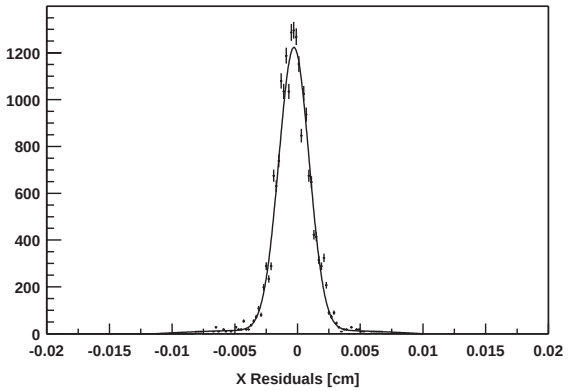


Fig. 13. Residual distribution of 1-pixel clusters in one chip, for the 50 μm pitch direction.

measured value of the resolution is above the theoretical value of $25\,\mu m/\sqrt{12} = 7.2\,\mu m$, where 25 μm is half of the pixel width, which is consistent with an additional contribution of the diffusion width of the charge cloud in the order of 5–7 μm. [1]

4.3. Tracking performance

The tracks measured in the pixel telescope have been used to determine the position of the collision vertices. Fig. 14 shows the distribution of the longitudinal coordinate of the reconstructed vertices. The three Pb targets, as well as the cryostat exit window of the beam tracker, are clearly visible. The distributions for the three targets can be described by a convolution of the target density function and the gaussian resolution function of the vertex reconstruction. Deconvolution leads to the measured thickness of the targets and to the reconstruction resolutions listed in Table 1.

Due to multiple scattering and increasing extrapolation distance, the reconstruction error increases from $\sim$ 200 to $\sim$ 500 μm when going to the most upstream target. The target thicknesses extracted from the deconvolution are in good agreement with the actual values of 1.5, 1.0 and 0.5 mm. The precision of the vertex reconstruction

[1] The value $d/\sqrt{12}$ holds true only for an ideal box distribution. However, in the present case the distribution gets smeared due to the finite diffusion width of the collected charges.

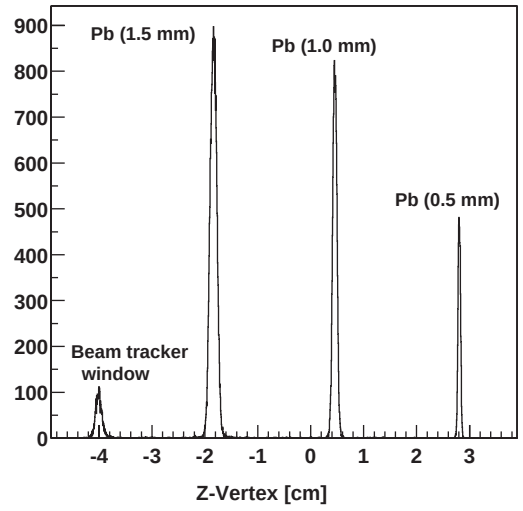


Fig. 14. Z positions of the vertices as measured with the pixel telescope. The peaks correspond to the Pb targets and material from the beam tracker cryostat window.

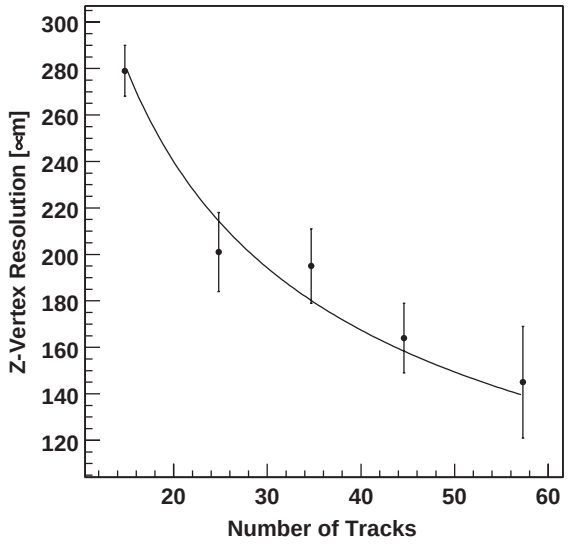


Fig. 15. Z-vertex resolution versus the number of tracks associated with the vertex, for the most downstream target.

Table 1
Z-vertex resolutions and thicknesses of the targets determined from the measured Z-vertex distribution

	Target 1	Target 2	Target 3
Z resolution	0.51 ± 0.01 mm	0.32 ± 0.01 mm	0.21 ± 0.03 mm
Thickness	1.49 ± 0.01 mm	0.98 ± 0.01 mm	0.48 ± 0.02 mm

depends on the number of tracks associated with the vertex, as shown in Fig. 15 for the most downstream target. The plot shows the Z-vertex resolution versus the number of tracks. The data points were fitted with a function of the form

$$\sigma = \frac{\sigma_0}{\sqrt{N_{\text{Tracks}} - 1}} \quad (1)$$

resulting in the value $\sigma_0 = (1050 \pm 30)\,\mu\text{m}$.

Fig. 16 shows the correlation between the x transverse coordinate of the vertex as measured in the pixel telescope and the same coordinate determined with the beam tracker. The correlation width is approximately 30 μm, including the tracking resolution of the beam tracker and the vertex resolution of the pixel telescope. Knowing that the beam tracker gives the transverse coordinates of the interaction point with a resolution

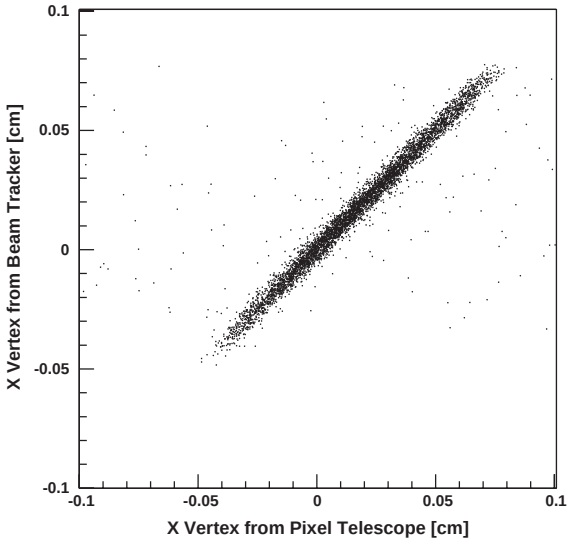


Fig. 16. Correlation between the x transverse coordinate of the vertex measured with the beam tracker and the same variable measured by the pixel telescope.

of ~20 μm, one obtains a transverse vertex resolution of the pixel telescope of approximately 20 μm, for minimum bias collisions. Like the Z-vertex resolution, the transverse coordinate resolution improves for the most central collisions.

5. Summary

The silicon pixel tracking telescope is a crucial detector of the NA60 experiment. In this paper we described the basic concepts determining the detector design, explained the assembly procedure and gave some results from lab tests of chip assemblies and modules. Three 4-chip tracking planes were used in October 2002 to study low energy Pb–Pb collisions. This was the first time that such pixel detectors, based on the Alice1LHCb readout chip, were operated in the high multiplicity environment of heavy-ion collisions. The measurements show that these pixel planes perform extremely well. In the meantime, the full telescope has been completed, and was successfully operated with high energy indium-indium collisions. The corresponding data analysis is ongoing.

Acknowledgements

We would like to express our gratitude to all the people who helped realising this challenging detector on a short time scale. This detector has been developed within the framework of the NA60 experiment. It is therefore natural to start by thanking our colleagues from NA60. In particular, we express our gratitude to D. Marras for help with the electronics aspects of the project. We very much appreciate the crucial contribution from the EP/ED and EP/MIC groups, and from the ALICE pixel project team, especially M. Campbell, P. Riedler, G. Stefanini and K. Wyllie. We also thank E. David, C. Joram, L. Kottelat, I. McGill and A. Onnela, of the EP/TA1 group, for their invaluable help in the design and construction of the pixel telescope. We are also indebted to R. de Oliveira, C. Millerin and M. Sanchez, of the EST/DEM group, for their contribution to the layout and production of the hybrid circuits.

References

[1] A. Baldit, et al., NA60 Proposal, Study of prompt dimuon and charm production with proton and heavy ion beams at the CERN SPS, CERN/SPSC 2000-010, SPSC-P316, March 2000.
[2] K. Wyllie, et al., A pixel readout chip for tracking at ALICE and particle identification at LHCb, Proceedings of the Fifth Workshop on Electronics for LHC Experiments Snowmass, Colorado, 1999, pp. 93–97.
[3] J.J. van Hunen, et al., Irradiation and SPS Beam Tests of the ALICE1LHCb Pixel Chip, CERN-ALI-2001-015, 2001.

Available online at www.sciencedirect.com
SCIENCE DIRECT®
Nuclear Instruments and Methods in Physics Research A 546 (2005) 51–55
NUCLEAR INSTRUMENTS & METHODS IN PHYSICS RESEARCH Section A
www.elsevier.com/locate/nima

The NA60 silicon vertex spectrometer

K. Banicz[a,*], A. David[b], M. Floris[c], J.M. Heuser[d], M. Keil[e], C. Lourenço[e], H. Ohnishi[d], E. Radermacher[e], R. Shahoyan[b], G. Usai[c]

[a]*Universität Heidelberg, Heidelberg, Germany*
[b]*Instituto Superior Técnico, Lisbon, Portugal*
[c]*Università di Cagliari and INFN, Cagliari, Italy*
[d]*RIKEN—The Institute of Physical and Chemical Research, Wako, Saitama, Japan*
[e]*CERN, 1211 Geneva 23, Switzerland*

Available online 31 March 2005

Abstract

The NA60 experiment studies the production of open charm and prompt dimuons in proton–nucleus and nucleus–nucleus collisions at the CERN SPS. To access the kinematics of charged particles already at the vertex level, a radiation tolerant silicon pixel detector has been placed in a 2.5 T magnetic field near the target. This vertex spectrometer was built from 96 ALICE1LHCB pixel chips arranged in 12 tracking planes.

The vertex spectrometer was successfully operated in a run with a 158 GeV/nucleon indium ion beam incident on indium targets in October–November 2003. During the five-week-long run it was exposed to high levels of radiation distributed inhomogeneously over the detector. The most exposed regions of the silicon sensors underwent type inversion.

With only a fraction of the total statistics analysed, the vertex spectrometer can already be seen to have dramatically enhanced the physics performance of NA60 with respect to that of its predecessors.

PACS: 29.40.Wk; 29.40.Gx

Keywords: Silicon pixel detector; Radiation damage

1. Physics program

The NA60 experiment studies extremely hot and dense matter produced in the collisions of ultra-relativistic heavy ions from the CERN SPS accelerator with a fixed target's nuclei. QCD predicts that above a certain temperature or energy density, a phase transition takes place from hadronic matter to Quark Gluon Plasma (QGP), in which quarks and gluons are no longer confined in colorless bound states. In order to gain

*Corresponding author. Tel.: +41 22 767 8670.
E-mail address: Karoly.Banicz@cern.ch (K. Banicz).

0168-9002/$ - see front matter © 2005 Elsevier B.V. All rights reserved.
doi:10.1016/j.nima.2005.03.013

information on this transition, NA60 analyzes the spectrum of the produced dimuons.

Near the phase transition chiral symmetry is expected to be gradually restored, possibly leading to observable changes of the shape of the ρ meson in the dimuon spectrum. For these changes to be seen, the resonances must be resolved at different collision centralities. This requires a good momentum resolution and high statistics.

The most direct evidence of the thermal equilibrium of the QGP would be the observation of thermal dimuons. In the intermediate mass region, the other known contributions to the dimuon spectrum are from the Drell–Yan process and the semileptonic decay of charmed mesons, which are long-lived enough to travel appreciable distances from the interaction vertex before decaying. The tracks of muons originating from charmed meson decays are thus displaced by a few hundred microns from the interaction vertex. Separating them from the prompt thermal dileptons calls for an excellent vertexing precision.

In the plasma the bound $c\bar{c}$ states (e.g., J/ψ or χ_c) should be dissolved, resulting in their suppression with increasing collision centrality. Counting them at different collision centralities demands high statistics.

2. Detector concept

Before hitting the target, the beam particles are tracked by a cryogenic silicon strip detector. This beam tracker measures the transverse coordinates of the interaction vertex with 20 μm precision. Placed in a 2.5 T magnetic field, the silicon vertex spectrometer tracks all produced charged particles within the muon spectrometer's acceptance. The hadron absorber can only be traversed by highly energetic muons, whose coordinates and momenta are measured by the muon spectrometer. Two muon spectrometer tracks pointing to the interaction region are required for the trigger. By matching the momentum vectors of the tracks in the vertex spectrometer with those in the muon spectrometer, we can identify which two of the hundreds of tracks in the vertex spectrometer were the muons. Measuring the charged particles' coordinates and momenta at the vertex and identifying the muons among them makes it possible to determine accurately the vertex position, to reject muons from kaon and pion decays, to measure the offset of the muons originating from charmed meson decays, and to refine the measurement of the muon momenta.

As dimuon production is a rare process, very high luminosity is needed to obtain high statistics. The high charged particle multiplicity of high-energy nuclear collisions imposes the use of high-granularity and radiation-tolerant silicon pixel detectors for the tracking in the interaction region.

3. The vertex spectrometer

The vertex spectrometer consists of eight 4-chip planes and eight 8-chip planes (Fig. 1). The 4-chip planes, which are in the upstream half, each cover the whole acceptance of the detector, whereas it takes a pair of 8-chip planes to do so. Altogether they give 12 space points for tracking. The first plane is at 7 cm from the target, the last one at 33 cm. The chips are glued onto a ceramic hybrid, whose back side is fitted with a cooling pipe, in which cold water is circulated.

Each pixel chip is an assembly of a sensor chip and a frontend readout chip. Both the sensor chip and the frontend readout chip have a matching 32×256 matrix of $425 \times 50\,\mu m^2$ pixels. The two are held together by 25 μm solder bumps connecting each sensor pixel with its corresponding readout pixel. The sensor chip is fabricated from 300 μm thick, high resistivity (15 kΩ cm) n-type bulk with p-type pixel implants. The 750 μm thick readout chip [1] is manufactured in 0.25 μm CMOS technology, and has an enclosed gate geometry and special guard rings, resulting in improved

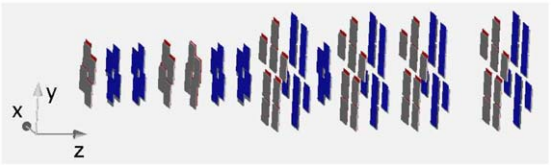


Fig. 1. The 96 chips of the vertex spectrometer arranged in 16 planes. The beam is in the direction of the Z-axis.

radiation tolerance [2]. It was designed by the CERN Microelectronics Group for the ALICE and LHCb experiments.

The detector is operated at 10 MHz. As the interactions occur randomly during 5 s long bursts, the 2-cycle-long trigger gate must be resynchronized to the clock. The 32 columns of a chip are read out in parallel, and the chips on a plane sequentially.

4. Operation

The detector was operated in 2003 in a 5-week-long run. During this time 6×10^{12} indium ions hit the 20% interaction length indium target. The resulting fluences were both simulated with FLUKA [3] and Venus [4], and measured during the first 3 weeks with *pin* diodes and Al activation. Fig. 2 shows the strong inhomogeneity of the radiation.

The temperatures and leakage currents were permanently monitored throughout the run. The total leakage current on a 4-chip plane increased typically from a few microamperes at the beginning to a hundred microamperes at the end of the run (Fig. 3). This corresponds to about 20 nA in the most exposed pixels, still well below the 100 nA limit of the preamplifier's leakage current compensation.

Radiation also decreases the effective doping concentration, and the n-type bulk eventually

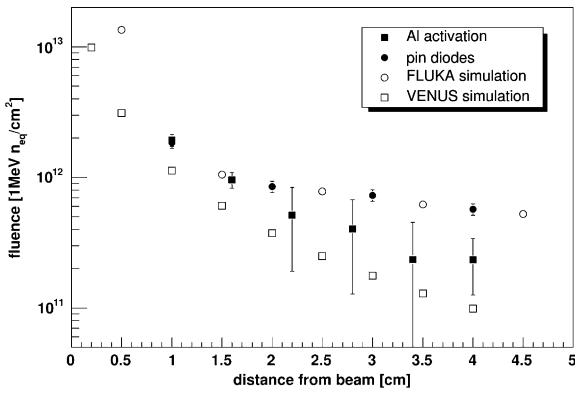


Fig. 2. The simulated and measured fluences 40 cm downstream of the target as a function of the radial distance from the beam.

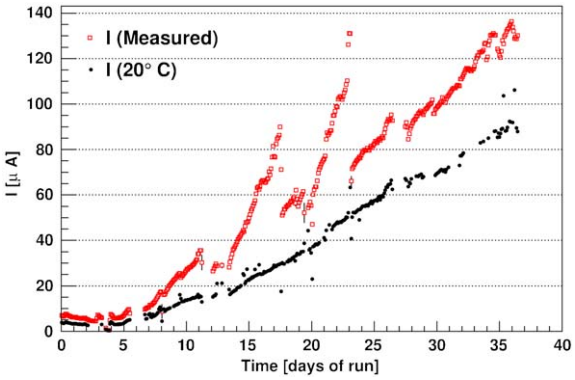


Fig. 3. History of the leakage current during the run on a 4-chip plane (squares). The spikes were caused by temperature rises due to deteriorating cooling, which needed to be repaired repeatedly. The solid circles show the leakage current scaled to a constant temperature.

becomes p-type. As a consequence, after type inversion the rectifying junction moves from the implants to the back plane, and the detector must be fully depleted in order to prevent the pixels from being short-circuited. Since after type inversion the depletion voltage increases with the fluence, pixels in the more exposed inner region need a higher bias voltage to function properly. This effect was observed in a bias scan in the fourth week of the run (Fig. 4).

In future, by use of these pixel chips the effect of the increasing leakage current will be kept under control by operating them at lower temperatures; however the rising depletion voltage will eventually exceed the device's breakdown voltage of about 300 V. This is expected to happen after an accumulated fluence of 4×10^{13} $1\mathrm{MeVn_{eq}/cm^2}$ if annealing effects are neglected. As a result an even larger central area will be left not fully depleted resulting in a loss of angular acceptance for the detector.

5. Performance

The vertexing precision in the Z coordinate (parallel to the beam) is obtained from the resolution of the targets' shape (Fig. 5). It is found to be about 300 μm, using on average fifty tracks to reconstruct a vertex. In the transverse plane the

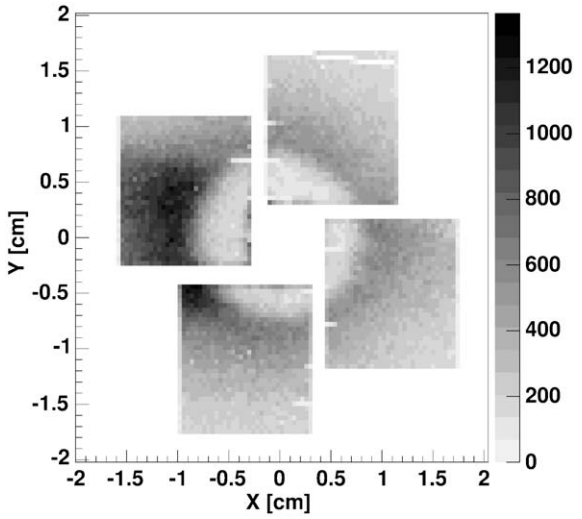


Fig. 4. Distribution of hits accumulated over 20 thousand events on a 4-chip plane during a bias voltage scan done in the fourth week of the run. The depletion voltage in the central part has already risen above the 30 V applied here, rendering the pixels inefficient in that area.

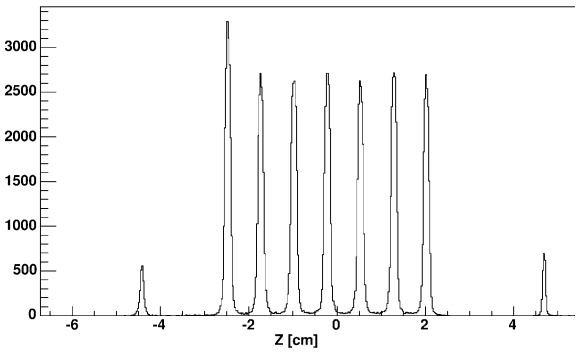


Fig. 5. The vertex distribution along the Z-axis. Visible are the thin vacuum windows of the target box and the seven indium targets, each a 1.5 mm thick cylinder. The front target diameter is larger than the others.

beam tracker already measures the position with a known resolution. From the comparison of the positions given by the the beam tracker and the vertex spectrometer, a resolution of 20 μm is inferred.

The vertex spectrometer measures the momenta of the muons before they suffer considerable energy loss and multiple scattering in the hadron absorber. Combining this information with the muon spectrometer measurement leads to a dimuon mass resolution unprecedented in heavy-ion collisions. (Fig. 6) [5].

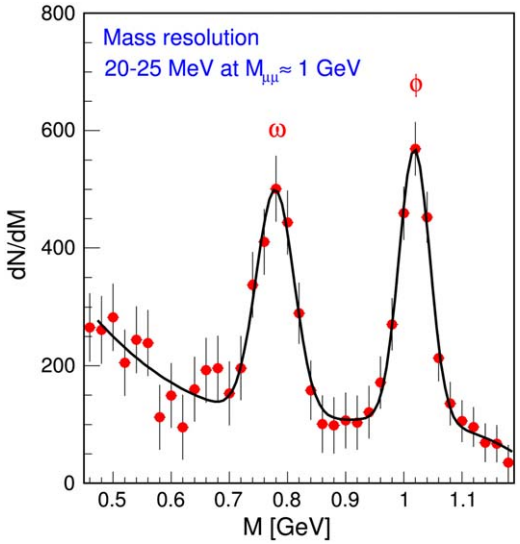


Fig. 6. The dimuon invariant mass spectrum in a small subsample of the indium–indium data after combining the measurements by the pixel and muon spectrometers, and subtracting the background. From the measured widths of the ω and ϕ resonances a resolution of 20−25 MeV is inferred. The resolution of the muon spectrometer measurement alone is ~80 MeV.

6. Summary

NA60 has successfully operated a silicon pixel detector in a high-rate and high-multiplicity environment. The detector has met the requirements of high speed, high granularity and radiation tolerance dictated by a broad and ambitious physics program. The chips have been exposed to fluences up to 5×10^{13} 1MeVn$_{eq}$/cm^2 and undergone type inversion. They are still operational and will be essential for the proton-nucleus physics run of 2004.

References

[1] K. Wyllie, presentation at this Workshop.

[2] W. Snoeys, et al., Nucl. Instr. and Meth. A 466 (2001) 366.
[3] A. Fassò, et al., eConf C0303241, MOMT004 (2003) [arXiv:physics/0306162].
[4] K. Werner, Phys. Rept. 232 (1993) 87.
[5] P. Sonderegger, et al., [NA60 Coll.], J. Phys. G 30 (2004) S1027;
H. K. Wöhri, et al. [NA60 Coll.], First results on Dimuon production from the NA60 experiment, Frascati Phys. Series 34 (2004) 185.
R. Arnaldi, et al. [NA60 Coll.], Rencontres de Moriond, QCD and Hadronic Interactions, 2004, hep-ex/0406054, to be published.

Appendix B

DAQ Technology in NA60

This appendix collects four publications related to the NA60 DAQ and, in particular, its hardware. It is a very important complement to chapter 4.

Section B.1 outlines the NA60 DAQ architecture. The first part deals with the software, while the second part describes data formats and hardware related issues.

Section B.2 details the influence of the PC architecture in the performance of CPU initiated PCI reads. It also provides exhaustive information on the tests performed in order to maximise the PCI read speed.

Section B.3 details the concept, architecture, performance and applications of the PCI FLIC card, while section B.4 provides similar information for the PCI CFD card.

The NA60 Experiment Readout Architecture

Michele Floris, Davide Marras, Gianluca Usai, André David, Peter Rosinsky, and Hiroaki Ohnishi

***Abstract*—The NA60 experiment was designed to identify signatures of a new state of matter, the Quark Gluon Plasma, in heavy-ion collisions at the CERN Super Proton Synchroton. The apparatus is composed of four main detectors: a muon spectrometer (MS), a zero degree calorimeter (ZDC), a silicon vertex telescope (VT), and a silicon microstrip beam tracker (BT). The readout of the whole experiment is based on a PCI architecture. The basic unit is a general purpose PCI card, interfaced to the different subdetectors via custom mezzanine cards. This allowed us to successfully implement several completely different readout protocols (from the VME like protocol of the MS to the custom protocol of the pixel telescope). The system was fully tested with proton and ion beams, and several million events were collected in 2002 and 2003. This paper presents the readout architecture of NA60, with particular emphasis on the PCI layer common to all the subdetectors.**

***Index Terms*—Data acquisition, heavy ion physics, PCI architecture, readout electronics.**

I. Introduction

THE NA60 experiment at the CERN Super Proton Synchroton [1] studies the dimuons produced in collisions of proton and ion beams on nuclear targets.

NA60 evolved from a series of previous experiments (the last one being NA50 [2]) from which it inherits some detectors. NA60 will improve some of the measurements made by these experiments and will complement them with some new measurements, thanks to a renewed experimental apparatus. In particular, NA50 observed an excess of dimuons in the mass region between the ϕ and the J/ψ (intermediate mass region), with respect to the expected sources (*open charm* and Drell–Yan). To clarify the origin of this excess it is necessary to separate the prompt dimuons from the muons resulting from the decays of *open charm* mesons. This requires precise tracking in the vertex region, which is achieved in NA60 thanks to state-of-the-art silicon detectors inserted in a 2.5 T dipole field, placed in the target region.

Even if the dimuon excess is seen to be due to other sources, such as the production of thermal dimuons, another signature of quark gluon plasma (QGP) production, the measurement of open charm production remains very relevant in itself and as the best normalization reference in the studies of charmonium suppression in nuclear collisions.

Manuscript received May 31, 2003; revised January 20, 2004.

M. Floris and G. Usai are with the Università degli Studi di Cagliari, 09100 Cagliari, Italy, and also with INFN, Sezione di Cagliari, 09042 Cagliari, Italy (e-mail: michele.floris@ca.infn.it).

D. Marras is with INFN, Sezione di Cagliari, 09042 Cagliari, Italy.

A. David is with IST, 1049-001 Lisbon, Portugal, and also with CERN, CH-1211 Geneva, Switzerland.

P. Rosinsky is with CERN, CH-1211 Geneva, Switzerland.

H. Ohnishi is with RIKEN, Wako-shi, Saitama 351-0198, Japan.

Digital Object Identifier 10.1109/TNS.2004.828828

Moreover, the much improved mass resolution achieved thanks to the tracking in the vertex region, allows a detailed study of the low mass resonances ρ, ω, ϕ, complementing measurements done by previous experiments [4], [3].

The readout of all detectors is based on the PCI architecture: the detectors are directly interfaced with the acquisition PCs via a set of custom electronics cards. In NA50, the readout was based on VME/CAMAC. The choice to move to PCI for all the detectors is justified by the higher performances and lower cost of a PCI-based readout system. This scheme turned out to be very flexible, allowing to replace the old VME/CAMAC-like readout systems for the MS and to implement the complex custom system of the pixel detector with the same PCI card. Moreover, as it will be clear in the following sections, the readout scheme which has been implemented simplifies the interfacing with the DAQ software, as the interface is the same for all the detectors.

In the following sections, we will first describe the experimental apparatus, then the data acquisition (DAQ) system following a top-down approach: from the DAQ software to the interfacing with the front ends.

II. Detector Concept

The NA60 experiment measures dimuons produced in proton-nucleus or heavy-ion collisions. By measuring dimuon production, the experiment studies the production of vector mesons, the Drell–Yan process, thermal dimuon production and, for the first time in heavy-ion collision, the production of charmed mesons. Moreover, it is very important to measure the centrality of the nuclear collisions, in order to compare the more elementary-like peripheral collisions with the more violent central collisions, where new physics is expected to show up.

The identification of events where pairs of D mesons are produced is done through the measurement of the impact parameter of the muon tracks, i.e., the distance in the transverse plane between the extrapolated muon tracks and the interaction vertex, at the point along the beam axis where the collision took place. This measurement requires a very precise vertex identification, which is achieved with the new silicon detectors placed in the vertex region.

The NA60 apparatus, as can be seen in Fig. 1, is composed of the following main detectors: a muon spectrometer (MS), a zero degree calorimeter (ZDC), a vertex telescope (VT), and a beam tracker (BT), which are briefly described in the following sections.

A. Muon Spectrometer

The MS was built for the NA10 experiment. It is composed of eight multiwire proportional chambers (MWPCs) for pre-

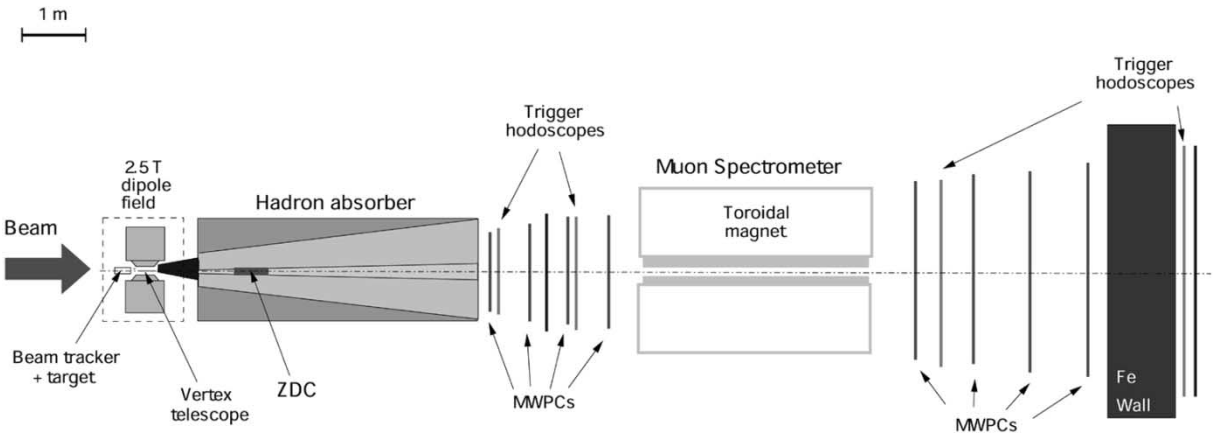


Fig. 1. The NA60 apparatus. The heart of the detector is the MS made by MWPCs, trigger hodoscope stations and a toroidal magnet. The ZDC is along the beam line inside the hadron absorber. The detectors installed in the target area are shown in an expanded view in Fig. 2.

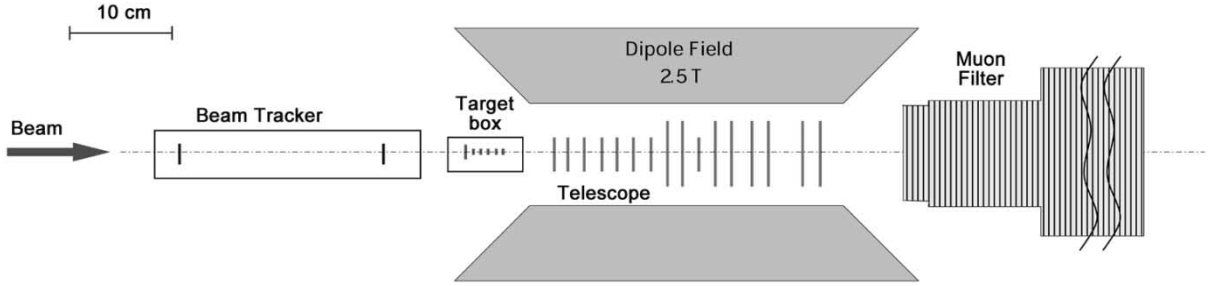


Fig. 2. Target area detail. This region contains the new NA60 silicon detectors: the BT (downstream the target) and the VT (upstream the target, inside the dipole field). Also note the segmented target which allows to have a bigger interaction length minimizing reinteractions.

cise tracking of muons and four hodoscopes of plastic scintillator slabs that provide the trigger signal for the experiment. In between the first four and the last four tracking stations sits a toroidal magnet, needed to measure the momenta of the muons. The last trigger hodoscope sits behind an iron wall, placed after all the chambers, to ensure that only muons can trigger the experiment without degrading the tracking resolution.

The MS is separated from the target region by a hadron absorber, to ensure that the abundantly produced hadrons will not pollute the trigger hodoscopes and tracking chambers.

B. Zero Degree Calorimeter

The ZDC is used to determine the centrality of the events by measuring the forward energy of non interacting beam nucleons. It exploits the Cherenkov light produced by charged particles cascading into quartz optical fibers, the sensible element of the detector. This detector has been inherited from NA50.

C. Vertex Telescope

The main challenge of the NA60 experiment is to match the muon tracks seen in the MS to the corresponding tracks measured right at the vertex, so as to eliminate the deterioration effects induced by the multiple scattering and energy loss suffered by the muons while crossing the hadron absorber. This requires a very accurate measurement of the charged particles, in the vertex region, a task accomplished by the silicon VT (Fig. 2).

While a silicon microstrip telescope is enough to do the job in the case of proton-nucleus collisions, the hundreds of charged particles produced in heavy-ion collisions would lead to an occupancy so high that the tracking would be impossible.

This imposes the exclusive use of truly bidimensional silicon pixel detectors, given their excellent granularity. Furthermore, since NA60 looks for very rare events, identified thanks to the very selective dimuon trigger, it must work with rather high interaction rates, unlike other ("minimum bias") heavy-ion experiments. This imposes the use of radiation tolerant pixel detectors, a technology that only recently became available and which is being used by NA60 for the first time.

In view of delays in the availability of the pixel detector, during the June 2002 proton run a silicon microstrip detector was used. Since this detector has a small material budget and covers the muon angular acceptance even at 40 cm away from the target, a dimuon mass resolution even better than with pixel could be obtained. Nevertheless, because of its low granularity, its use is restricted to proton runs. Moreover, subsequent Monte Carlo simulations have shown that the best setup for proton runs is an hybrid telescope made by both strip and pixel planes. This hybrid setup will be used for the 2004 proton run.

D. Beam Tracker

Upstream of the target, a BT [5] (Fig. 2) measures the flight path of the incoming beam particles and determines the transverse coordinates of the interaction point, at the target, with a precision of around 20 μm. These microstrip silicon detectors operate at a temperature of 130 K, in order to increase their lifetime in the extreme radiation conditions of the experiment.

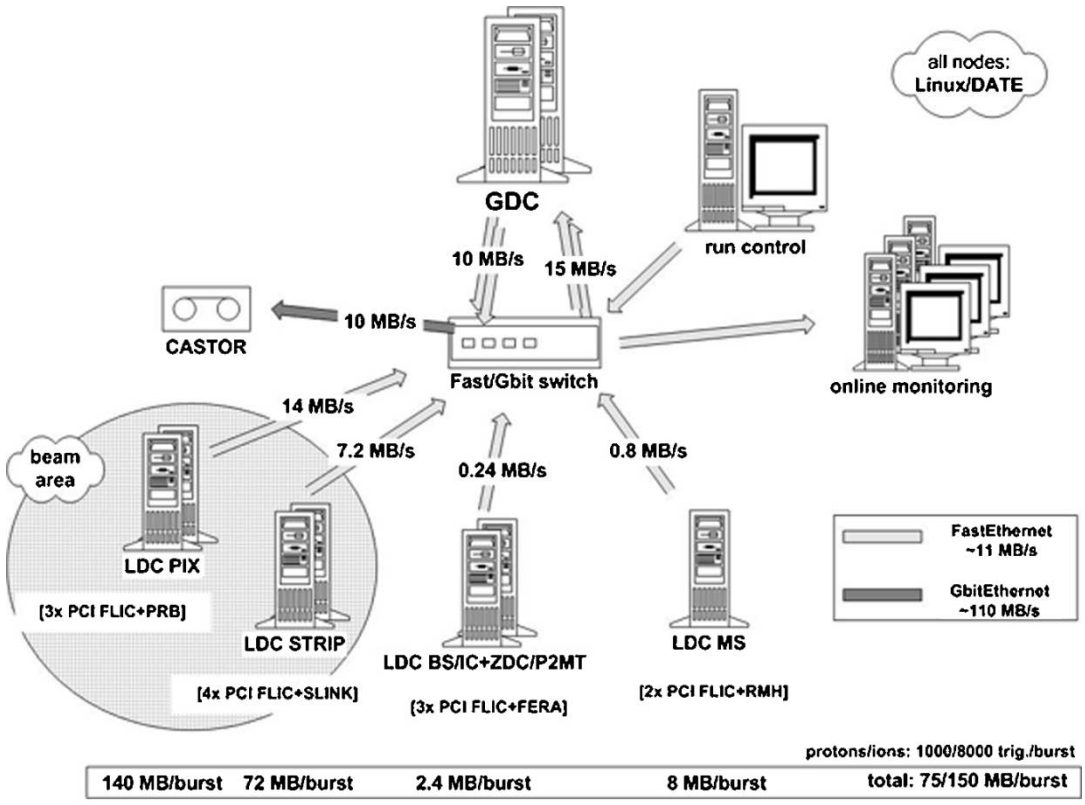


Fig. 3. DAQ system architecture. Data throughput is referred to ion runs, unless indicated otherwise.

III. DAQ Architecture

At the CERN SPS particles do not arrive on the target continuously, but rather within limited time intervals called "bursts" (typically 5 s long in proton runs and 6 s long in ion runs). The period of time in between two bursts is called "interburst" (typically 10 s long).

NA60 has adopted a scalable philosophy for its DAQ system (Fig. 3). Each detector is divided into one or more "partitions." During the burst, when events are produced and detected, each partition is read out independently by the electronics, whose last stage is a general purpose PCI card (see Section IV-A). This PCI card is the same for almost every detector (with the single exception of the microstrip vertex detector [6]). It has a large local memory buffer (64 MB) where data are stored during the burst. These cards sit in commodity PCs which are called local data concentrators (LDC).

The readout cycle is started by a trigger signal produced by some special logic which gets the data from the MS hodoscopes and looks for patterns compatible with the presence of a dimuon inside the acceptance of the spectrometer.

During the interburst data are read from the local memory of the various PCI cards and stored on the main memory of the LDCs. Data from the various LDCs are then collected and put together ("*built*") by a single PC, called global data concentrator (GDC). The GDC sends the global event to CERN's Central Data Recording Facility (CASTOR [7]) for permanent storage on tapes.

A. DAQ Software

The NA60 DAQ software is based on the data acquisition and test environment [8] (DATE) framework, developed by the ALICE collaboration.

This software is designed to run as a distributed system of several hosts performing the tasks of readout (LDC), event building (GDC), run control (RC), online monitoring (MON), and message handling (INFO). The operating system of DAQ processors must be UNIX-like, they must share the same filesystem and must be connected by TCP/IP network. The DAQ chain in DATE (Fig. 4) works as follows: the detector-specific program *readout* running on LDC collects the data from the front-end electronics (from the PCI card buffer, in the case of NA60) and stores them in a local buffer. The buffer is shared with the *recorder* process that off-loads it, asynchronously sending the data over the network into the GDC. Here, the gdcServer (one instance per LDC) stores the received data into another buffer that is off-loaded by central *eventBuilder* process. The *eventBuilder* waits until all the subevents from the LDCs arrive, builds the full event (adding a special header), performs consistency checks and writes the event into a file. In a multi-GDC environment the events are distributed among the GDCs to share distribute the load. The adopted two-stage buffering scheme naturally balances the load in the system and decouples the event building from the readout stage. The whole system is controlled from a single point: the *runControl* process on RC host opens sockets to the DAQ hosts, where certain Internet daemons are running. On LDC, the *rcServer* daemon controls the *readout* and

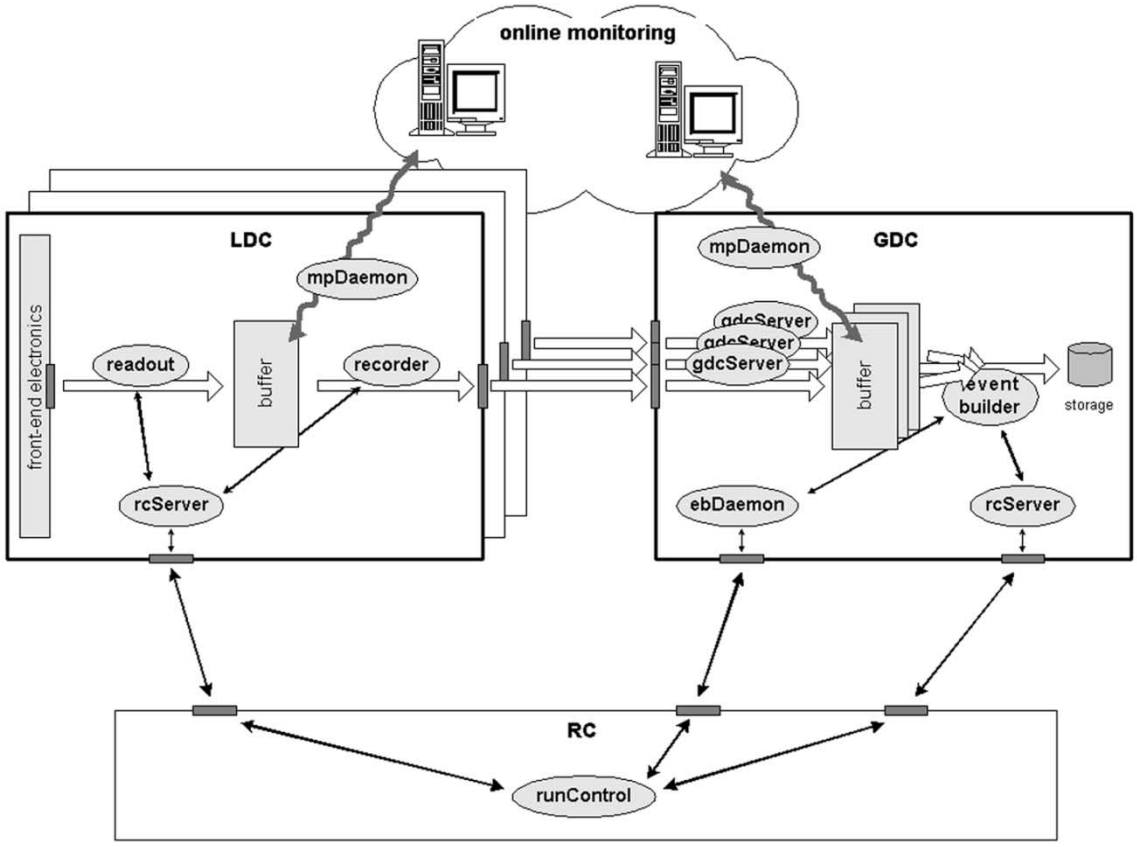


Fig. 4. DAQ, RC, and monitoring in DATE.

recorder processes, on GDC, the *rcServer* and *ebDaemon* daemons control the *gdcServer* processes and central *eventBuilder*. To facilitate the online monitoring, a dedicated *mpDaemon* is started upon request and sends the events from the LDC or GDC buffer to the monitoring workstation: a fraction of events is sent from LDCs to the monitoring hosts to check the functionality of the detectors and the data quality during the runs. Each DAQ host can generate information or error messages that are collected by *infoLogger* host by *infoDaemon* Internet daemons (one per connection).

The NA60 DAQ system consists of central run control (RC) host, one or more LDCs for detectors (1xBT, 1xZDC, 1xMS, and 3xVT [1]), one special LDC (reading nondetector information), three GDCs and several monitoring (MON) hosts (Fig. 3). The RC host contains also the DAQ software repository and exports it over NFS to all DAQ machines. It also serves as the infoLogger host and main control console.

Fig. 3 also shows the data throughput expected: up to 150 MB/burst are expected in ion runs and this results in a sustained flow of about 10 MB/s into tapes. The data rate is highly dominated by the VT (Section IV-E).

IV. The PCI System

As aforementioned, the basic unit of the PCI system is a general purpose PCI card. Each detector is interfaced to this card via a detector specific mezzanine, which is in general different for each detector. Here, we will first describe the general purpose PCI card, then the particular implementation for each detector in the experiment.

A. PCI Card

The NA60 readout was initially based on the PCI-FLIC [9] (FLexible Input/output Card), developed by the EP/ED-DTb group at CERN. This card contains a programmable logic (FPGA) with an embedded hardware PCI core (ORCA OR3LP26) and a large memory buffer (32 or 64 MB). The FPGA was programmed using VHDL code and a synthesizer, plus a tool from Lucent for placement and routing. The development of the VHDL code was mainly done by the NA60 team.

This card has been successfully used in datataking runs in 2001/2002, validating the readout architecture.

Subsequently a new card [called compact and flexible design (PCI-CFD) Fig. 5] equivalent to the FLIC from the architectural point of view but with much faster logic (based on an ALTERA APEX EP20K100E FPGA) has been developed and adopted. Thanks to this new card the microstrip vertex detector will also be included in the readout scheme described here (see Section IV-E).

The PCI-CFD has been used in all datataking periods in 2003.

The interfacing with the PCI bus is handled via a PLX 9030 PCI bridge, which allows the user to program transactions on a local TTL bus, in a very simple way. This chip can be highly customized, and the customization values are written on a EEPROM directly from the PCI bus using a Linux program. Using an external component as a PCI to local bus bridge decouples the problems of FPGA design development from the

[1]In June 2(144 roton run 4 LDCs were used for the microstrip VT.

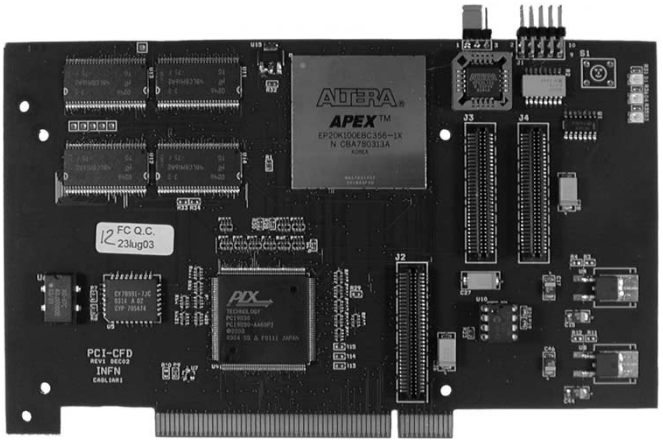


Fig. 5. PCI-CFD general purpose PCI card. This card is the basic readout unit of the NA60 experiment. It contains an Altera programmable logic, a PLX PCI bridge and a large memory buffer.

PCI interfacing, simplifying considerably the card's debugging and thus leading to a design which is more reliable and easier to maintain. In the PCI-FLIC the hardware PCI interface was embedded in the FPGA.

The PCI-CFD contains a large amount of memory as well (64 MB) and the connectors to plug mezzanine cards. However this card is not fully compliant with the IEEE P1386.1 standard for PMC mezzanine cards.

It is also equipped with a silicon serial number DALLAS DS2401, which permits to uniquely identify every PCI-CFD.

Both PCI-FLIC and PCI-CFD cards are designed to work as a PCI target, which allows to get more than satisfactory performances, with less problematics with respect to a PCI master: with an accurate choice of the host computer it is possible to reach a maximum bandwidth of almost 38 MB/s [10].

In the case of the PCI-FLIC the maximum measured bandwidth was some 15 MB/s. With the PCI-CFD the bandwidth is 30 MB/s; this number is not as dependent on the host computer as it was with the PCI-FLIC, where one could get 50%–60% differences in bandwidth between different computers.

Fig. 6 shows a block diagram and data flow of the whole application. The dashed line encloses blocks which are inside the FPGA. There are four main blocks (READOUT, EVENT FORMATTING, SDRAM CONTROLLER, and PCI INTERFACE), plus a CONTROL LOGIC, which triggers the other blocks, arbitrates accesses to memory, performs the handshaking with software and handles signals from the experiment (like trigger, burst, and busy signals).

Fig. 7 shows how the handshake between software and hardware is implemented. The software is continuously polling a status register implemented in the PCI card. The lowest two bits of this register contain the SPS (BURST/INTERBURST) status, while the third one (bit 2) is used to implement the handshake itself. When the burst is over and the last event has been collected the hardware sets bit 2. As soon as the software sees this bit it starts to move the data resetting it at the end. If a new burst arrives before bit 2 is reseted, bit 3 is set (by the hardware) indicating a timeout.

The READOUT block gets the data from the detector via the mezzanine. This block is detector specific and depends on

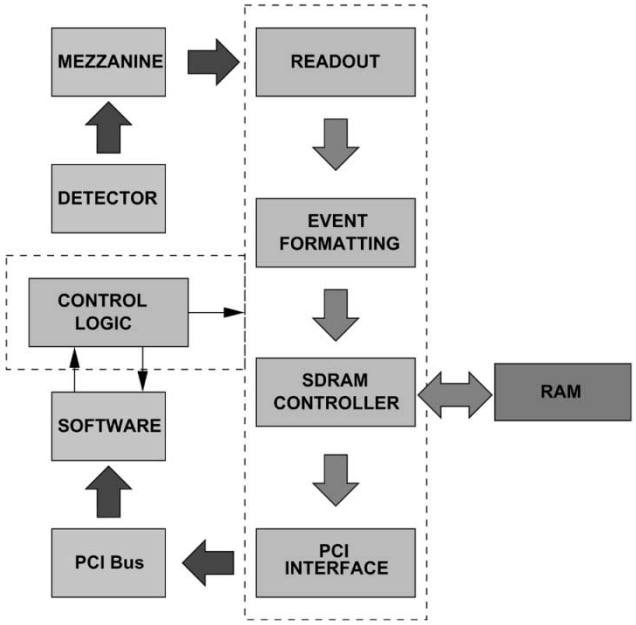


Fig. 6. Application block diagram. The dashed line encloses blocks implemented inside the FPGA.

the mezzanine. It will be discussed in detail in the following sections.

The EVENT FORMATTING block frames the incoming data within a header and trailer on an event by event basis. The header and trailer contain all the information necessary to build the "global" event, like number of data and total words, number of burst, number of event inside the burst, time of arrival of the event, and error flags. The data format is depicted in Fig. 8.

The SDRAM CONTROLLER writes the formatted events into the memory. During the interburst it is connected to the PCI Interface to send data from the on-board memory to the main LDC memory (see Section IV-A).

B. Linux Driver for the PCI Cards

The operating system used for the DAQ hosts is Linux. The version used is CERN Linux 6.1.1, which was the CERN standard when the project started (2001). This version is based on a kernel release 2.2.x.

A driver for both cards, FLIC and CFD, has been developed for this operating system. It hides the hardware peculiarities of the PCI card (like absolute offset of its internal registers and memory), making it available as a logical entity, through an entry in the `/dev/` directory in the Linux tree. Through the driver one can simply access the card using the C call `open("/dev/cfd0",O_RDWR)` and using some macros that define the local offset of memory areas inside the card, instead of opening the `/dev/mem` device and accessing memory areas using absolute offsets.

It also sets all the important environment parameters needed to achieve the best possible performances (i.e., the `mtrr` registers which allows caching and burst transfers [10]).

Finally, the driver allows to get some parameters from the PCI card through the Linux `ioctl( )` system call, like the unique identifier (Section IV-A) or the number of detected cards in the system.

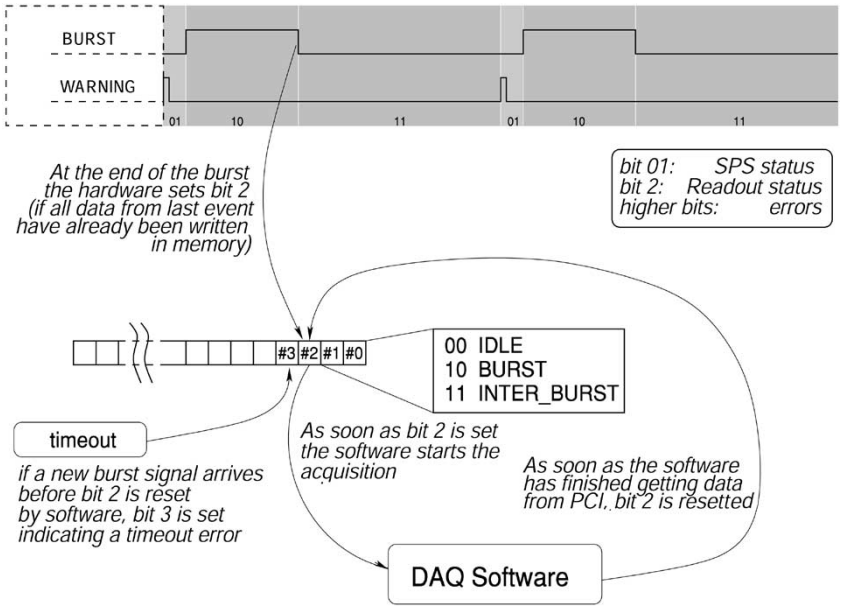


Fig. 7. Readout cycle and handshake between software and hardware.

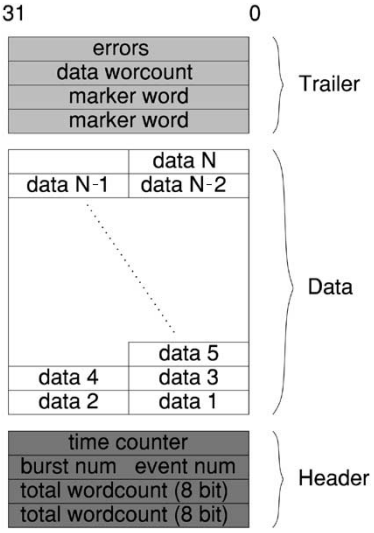


Fig. 8. NA60 data format. The header and trailer are common to all detectors. The "data" words framed between header and trailer are encoded in a detector-specific format.

C. MS

MS data (both chambers and hodoscopes) are read and temporarily stored by a system of CAMAC modules developed at CERN in the mid 70 s, called receiver memory hybrid (RMH) [11]. These modules can contain only one event and must then be read before a new trigger can be accepted.

The total number of readout channels of the MS (MWPCs + hodoscopes) is about 20 000, giving an average event size of about 1 kB.

In the previous experiments data were read by a custom VME module (called MEMRMH) which contained a small memory buffer (4 MB), limiting the number of events which was possible to acquire in a single burst to about 4000. This VME module was connected to a VAX-based DAQ through a system of 146puters.

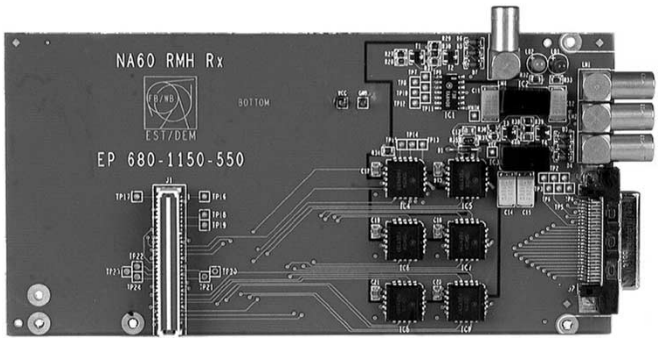


Fig. 9. The MS RMH mezzanine. This mezzanine is "passive" and converts incoming signals to TTL, so that they can be used inside the PCI card. All the readout logic is implemented in the PCI card.

The new readout based on PCI does not have this limitation anymore, and the number of events which is possible to acquire is limited only by the dead time of the detector, which has also been reduced and is at present around 120 s on average (mainly due to the CAMAC system). The MS is the slowest detector in the experiment.

The mezzanine used for this detector (called RMH mezzanine, Fig. 9) is very simple. It converts the incoming signals from ECL and NIM, used by the font-end electronics to TTL. The protocol is implemented inside the READOUT block in the FPGA of the PCI card.

The RMH protocol is a double handshake protocol, similar to the VME one. Fig. 10 shows a typical RMH cycle. The PCI card is the master and asserts a "START_READ" signal which frames the whole transaction and then asks for the first data word, asserting the "ENCODE" signal. The CAMAC modules, which are the slave, respond asserting the "DFLAG" signal when valid data are on the bus. When no more data are available for reading, the slave asserts the END_OF_READ signal.

Two PCI cards with RMH mezzanine are needed to read the MS (one for MWPC and one for hodoscopes) but they share the same LDC.

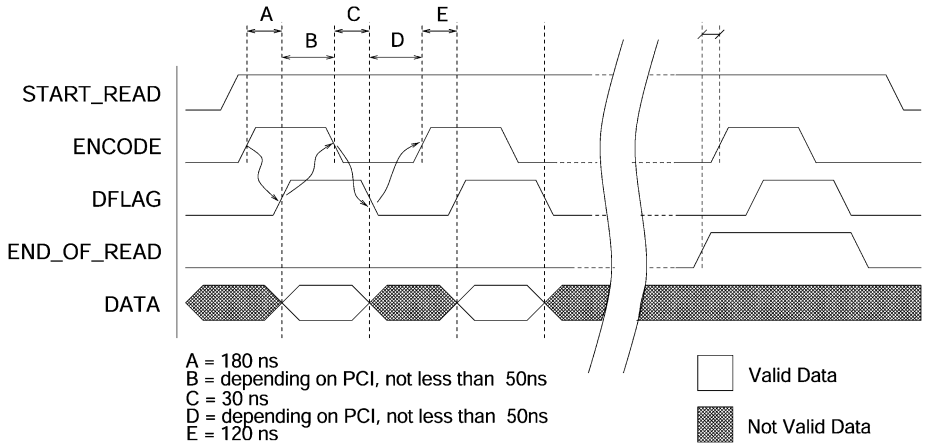


Fig. 10. RMH protocol cycle.

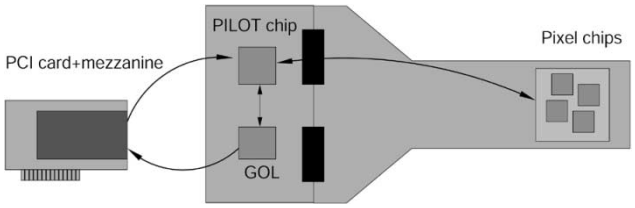


Fig. 11. Pixel readout system layout. The pixel chips are read out by the pilot chips which sends the data to the mezzanine through GLink, via the GOL chip.

D. ZDC and BT

The ZDC and BT readout applications are similar to the MS one and use the same mezzanine (RMH). In these cases, it is necessary to read a FERA CAMAC module.

This protocol is very similar to the RMH one (double handshake), the main difference being that the PCI system is now the slave.

The average event size is less than 1 kB for both the ZDC and the BT.

Although from the point of view of the performances these two detectors could be read using the same LDC, it has been decided to use two distinct LDCs for practical reasons.

E. VT

The microstrip VT has a slightly different readout architecture [6], based on a different PCI card and it will not be discussed here.

The silicon pixel telescope, on the other hand, represents the most sophisticated application.

Fig. 11 shows the pixel detector readout system. The detector is composed of 16 pixel planes based on the ALICE1LHCB [12] readout chip (pixel chip in the following) developed in the framework of the ALICE and LHCb collaborations at CERN. This chip is a radiation tolerant 32×256 matrix of readout cells, each 425×50 μm^2. Its operation is highly customizable: several internal DACs can be set via a JTAG interface [13].

Each plane contains four or eight pixel chips, bringing the total number of channels in this detector up to about 720 000. The expected maximum occupancy is about 3%–4%, giving a maximum event size of about 18 kB. Assuming a maximum number of triggers of 8000 the corresponding data throughput

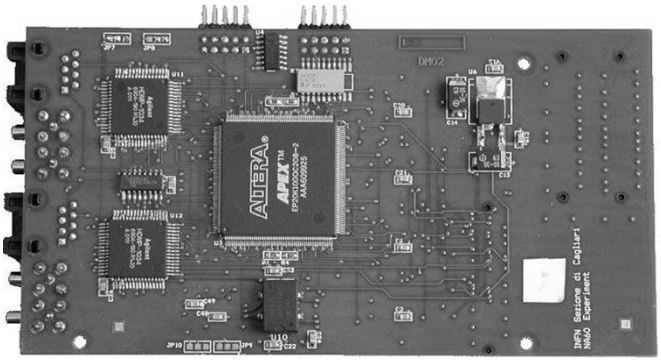


Fig. 12. Pixel readout mezzanine. This mezzanine contains a programmable logic, which is used to implement the readout logic, and some FIFOs which are used to temporarily store the events.

is 140 MB per burst. The VT produces more the 90% of data in each burst (Section III-A).

Pixel chips are clocked at 10 MHz. During event readout 32 bit pixel rows (256 per chip) are extracted in subsequent clock edges, so that the readout time is 25 s per chip. Pixel chips sitting on the same plane are read out sequentially. Reading out four chip planes therefore takes around 100 s while reading out eight chip planes takes around 200 μs. The dead time is reduced by a multi-event buffer present inside the pixel chip, which allows to accept up to three new events during the readout stage. The presence of the multi-event buffer and the partitioning of the detector lower the effective dead time below the one of the MS (120 μs, Section IV-C).

The pixel chip is interfaced over a data/control GTL bus to a second radiation tolerant chip (the PILOT chip [14]). This chip on one hand converts the GTL levels to CMOS and distributes all readout control signals and JTAG commands coming from the counting room to the pixel chip. On the other it transmits the data downstream to a radiation tolerant serializer chip (the GOL [15]) which finally sends the data to the PCI system using the industry GLink protocol [16].

The pixel readout mezzanine, shown in Fig. 12, contains a FPGA which implements the communication with the PILOT chip through a serial LVDS line (JTAG commands, clock distribution and PILOT configuration commands) and gets the data via two HP 1034 GLink receivers from two differe 147 anes.

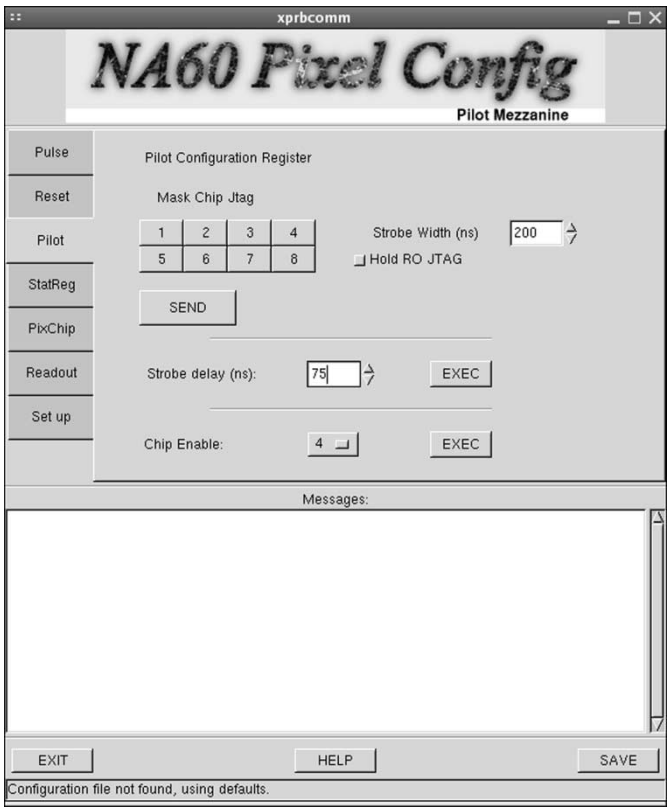


Fig. 13. The graphical user interface used to configure the pixel detector.

At this level data are zero suppressed, encoded and stored temporarily in a FIFO, which is subsequently accessed by the PCI card.

Commands are sent to the mezzanine, the PILOT chip and the pixel chip by a custom software which writes specific registers on the PCI card. The FPGA application of the PCI card in the case of the VT contains one extra block with respect to those depicted in Fig. 6, which acts almost independently and is used to send commands to the mezzanine through a serial link. Commands to be sent are written in a specific PCI register, then another register is accessed to trigger the transmission of the command. The mezzanine interprets the received command and acts accordingly (either it sets some internal parameters, either it forward a JTAG command to the pixel chip/PILOT chip system). Some commands produce an output which is read by the PCI card through the same FIFOs used for data and written in another PCI register. The software can access output resulting from certain commands through this PCI register (a typical example is the reading of a status register inside the mezzanine).

A complete software package has been developed for the Linux operating system which allows to easily configure the detector. It consists of two control programs, which are used respectively to configure internal parameters of the mezzanine (for example, it needs to know how many pixel chips are on the plane) and to send JTAG commands to PILOT chip and pixel chip. The JTAG controller is software-based: the hardware simply forward the bit sequence it gets from the software. This program is interfaced to a mysql database which contains the optimal settings for every pixel chip. Those two control programs are also used by DATE to configure the detector at run startup.

A graphical user interface (GUI, Fig. 13) has been developed for these programs, which allows to easily configure the detector.

Finally, a program performs a threshold scan of the detector, to find dead or noisy pixels and tests the performances of the detector.

V. DAQ Performances

The new readout system has been used in several different conditions over the past two years.

In October 2001 (with the PCI-FLIC) the PCI electronics for the MS was used for the first time, in a proton test run. During this run, we reached a trigger rate as high as 9000 triggers in a 5 s burst. This run was meant to commission the new MS readout electronics only.

In the June 2002 proton run, all detectors (except the microstrip vertex detector) were working with the PCI-based electronics (with the PCI-FLIC), though only one pixel plane was available. Tracking in the vertex region was assured by the silicon microstrip VT. 800 000 dimuon events were collected.

In the low-energy Pb run of October 2002, using the PCI-FLIC card, three pixel planes were read out with the new system, allowing to test a small pixel telescope. The MS was not used during this run, while the other detectors (BT and ZDC) were included in the acquisition and read out with the PCI system. Around 30 000 000 events were collected.

In August and September 2003, in two test beams with pions and protons at the CERN SPS, eight planes of the pixel detector were tested using the PCI-CFD on beam for the first time.

In the October 2003 indium run, all the detectors were present, including a complete silicon pixel telescope made of 16 planes. During this run the pixel telescope, the BT and the ZDC were read out using the PCI-CFD card, while the MS was read out using the PCI-FLIC card. The average event size of the pixel detector was about 10 kB, and the trigger rate was about 6000/7000 triggers/burst. For testing purposes, short runs were taken with a trigger rate of more than 9000 trigger per burst. Even if the DAQ can stand this trigger rate it was decided to run with 6000 triggers/burst to minimize the pileup rate.

VI. Conclusion

NA60 is one of the first high-energy physics experiments to use an architecture fully based on the PCI bus for its DAQ system. This bus offers attractive features, like high performances at a relatively low price and very easy interfacing with the acquisition software. Moreover, using PCI hardware cores and bridges eases the PCI card development.

The readout system described here has been successfully used during several datataking runs over the past two years.

Acknowledgment

The authors would like to thank the EP/ED—DTb group at CERN and, in particular, H. Muller for the technical support and the many ideas in the early stages of the project.

They would also like to thank the ALICE pixel team—in particular, F. Formenti, G. Stefanini, K. Wyllie, A. Kluge, M. Burns,

P. Riedler, and M. Campbell—for their constant support in the development of the NA60 pixel detector. A special acknowledgment goes, of course, to the entire NA60 Collaboration.

REFERENCES

[1] *Study of Prompt Dimuon and Charm Production with Proton and Heavy Ion Beams at the CERN SPS*, May 15, 2000. NA60 Proposal.

[2] *Study of Muon Pairs and Vector Mesons Produced in High Energy Pb-Pb Interactions*, Nov. 1991. NA50 Proposal.

[3] M. C. Abreu *et al.*, "Low mass dimuon production in proton and ion induced interactions at the SPS," *Eur. Phys. J.*, vol. C14, pp. 443–455, 2000. NA38/NA50 Collaborations.

[4] G. Agakishiev *et al.*, "Enhanced production of low mass electron pairs in 200-GEV/U S-Au collisions at the CERN SPS," *Phys. Rev. Lett.*, vol. 75, pp. 1272–1275, 1995. CERES Collaboration.

[5] P. Rosinsky, "Thecryogenic Beam Tracker of NA60 for heavy ion proton beams," *Nucl. Instrum. Meth. A*, vol. 511, no. 1–2, pp. 200–204, Sept. 2003.

[6] A. David *et al.*, "The readout system of the NA60 silicon microstrip tracker," *Nucl. Instrum. Meth. A*, pp. 118–126, Aug. 2003.

[7] J. P. Baud *et al.*. CASTOR status and evolution. presented at 2003 Conf. Computing in High Energy and Nuclear Physics. [Online]http://www.slac.stanford.edu.econf/C0303241/

[8] M. Arregui *et al.*, "The ALICE DAQ: Current status and future challenges," *Comput. Phys. Commun.*, vol. 140, pp. 117–29, 2001.

[9] H. Muller, F. Bal, A. David, D. Dominguez, and J. Toledo. A flexible PCI card concept for data acquisition: The PCI-FLIC. presented at 12th IEEE-NPSS Real Time Conf.. [Online]http://ific.uv.es/rt2001/

[10] A. David. (2001, June) Performance Issues in PCI Reading. [Online]. Available: http://adavid.home.cern.ch/adavid

[11] J. B. Lindsay, C. Millerin, J. C. Tarle, H. Verweij, and H. Wendler, "A fast and flexible data acquisition system for multiwire proportional chambers and other detectors," *Nucl. Instrum. Methods*, vol. 156, p. 329, 1978.

[12] K. Wyllie *et al.*. A pixel readout chip for tracking at ALICE and particle identification at LHCb. presented at 5th Workshop Electronics for LHC Experiments. [Online]http://lebwshop.home.cern.ch/lebwshop/LEB99_Book/LEB99_Proceedings.html

[13] *IEEE standard test access port and boundary-scan architecture*, IEEE STD. 1149.1a, 1993.

[14] A. Kluge. The ALICE on detector pixel PILOT system—OPS. presented at PIXEL 2002 Workshop. [Online] http://www.slac.stanford.edu/econf/C020909/

[15] P. Moreira *et al.*, "A 1.25 Gbit/s serializer for LHC data and trigger optical links," *Proc. 5th Workshop Electronics for LHC Experiments*, pp. 194–198, Sept. 20–24, 1999.

[16] Agilent HDMP-1032, HDMP-1034, Transmitter/Receiver Chipset [Online]. Available: http://www.semiconductor.agilent.com

Performance issues in PCI reading

A. David*

31st May 2001

NA60

Note

2001–3

Rev. 2

Abstract

This note describes the work developed in order to achieve the maximum throughput of 37.8 MB/s on CPU initiated PCI reads on a commodity x86 personal computer. After briefly presenting the PCI bus, discussing NA60's experimental requirements and read *vs.* write performance, we present the details of testing several hardware configurations. Next we give a quick checklist of the system where top PCI reading speed was achieved. Finally we draw upon the experience gathered to present some prospects and conclusions on the subject.

Contents

*IST, Lisbon, Portugal

1 Introduction

All data acquisition (DAQ) technology at CERN has been based on some or several proprietary or outdated busses like CAMAC, VME or FastBUS. These buses are known to require expensive custom modules and a significant amount of maintenance and knowledge. Since the time when these were "state of the art" for DAQ, computer architectures have evolved by several orders of magnitude, to such an extent that probably the graphics adapter on the computer you use today has a comparable storage capacity to the total storage on those times' systems.

This means that the everyday personal computer (PC) can easily outperform those custom-made systems, in both performance, price and maintenance, whilst still usable for offline processing of data – or writing technical notes.

Taking into account this state of affairs, a team from the Electronics Design group of the Experimental Physics division at CERN has developed the PCI Flexible I/O Card, PCI-FLIC. An extensive description of the FLIC, its internals and interfacing, can be found in [1].

The FLIC has been adopted by the NA60 experiment as the cornerstone for the readout systems of all its detectors. The first system leaving VME readout to be handled by a FLIC, is the RMH readout protocol used for the MWPCs and hodoscopes. At every trigger during an SPS burst, the FLIC will write the MWPCs and hodoscopes data to its internal SDRAM buffer. When the burst is over, data is transferred into the PC's main memory. Then the data acquisition and test environment (DATE) software framework transfers the data from each local data collector (LDC) PC to a global data concentrator (GDC) and from here to permanent storage. Document [2] describes DATE's functionality, which was developed for the ALICE experiment at CERN.

NA60's DAQ system is depicted in fig. 1 on the following page. The throughput values presented are for heavy ion runs, which represent the most data transfer demanding case.

In [3] it is described how the PCI-FLIC readout protocol is being implemented for NA60.

2 The PCI bus

The peripheral components interconnect (PCI) bus is nowadays the most common, if not the only expansion bus found in PCs. Besides common PCs based on x86 processors, one also finds PCI buses in all recent Alpha, Sun and PowerPC computers.

A commodity x86 PC usually has a 32-bit, 33 MHz PCI bus, although the PCI specification defines 32 and 64-bit modes of operation at clock frequencies from 0 to 66 MHz. 64-bit, 66 MHz versions of the PCI bus are mainly used for high-end servers needing fast networks links with bandwidths in excess of 100 MB/s, like Gigabit Ethernet.

The theoretical limit for the PCI bandwidth in a 32-bit, 33 MHz PCI bus is $\sim$126 MB/s. Now, this speed would be attained if data were transferred at every single clock cycle. But since one has:

- Protocol overhead,
- Several devices on the bus, and
- Each agent presents its own idiosyncrasies,

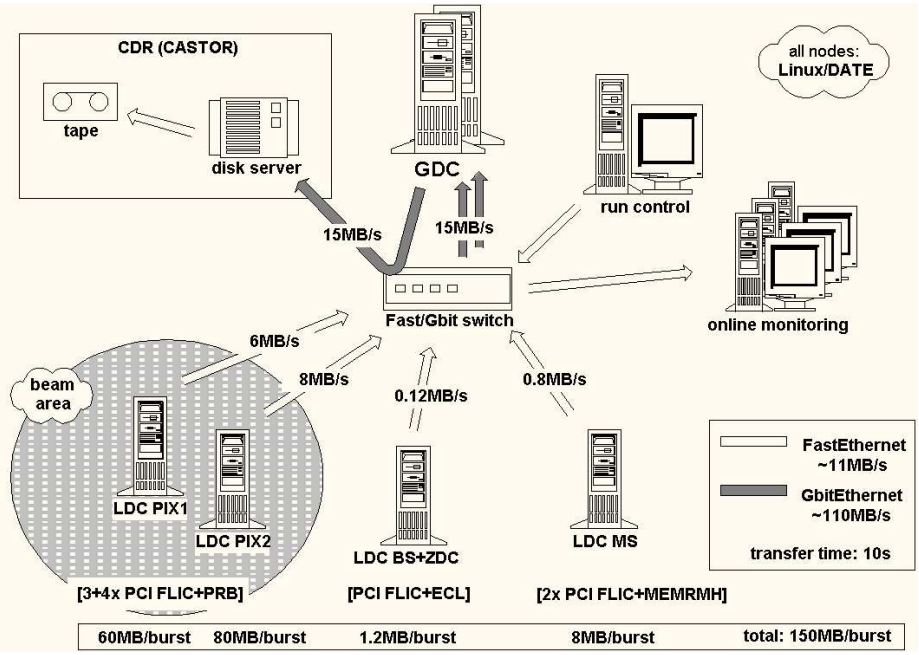


Figure 1: The NA60 DAQ network. Please refer to the main text for further information.

effective transfer values are always smaller.

A thorough explanation of the PCI bus and the way it works can be found in [4]. However it is useful to describe the way transactions happen:

- A PCI agent needing to initiate a transaction on the bus has to request bus ownership from the bus' **Arbiter**.
- Eventually, ownership is granted, and the agent being given the bus starts a transaction in which it'll be the **Master**.
- All agents in the bus listen for the master's request, and the one accepting the Master's request is the **Target** for that transaction.
- The target then complies to whatever was ordered by the master.

So, target-only agents are limited to exclusively being read from or written to, since these cannot initiate transactions. Agents with master capabilities can write to or read from target agents.

Having this intrinsic asymmetry on PCI operations, it is worthwhile to underline the natural differences in performance between reading and writing. This is crucial when discussing the performance for PCI reading.

2.1 Reading *vs.* Writing

The most simple read and write transactions on the PCI bus are single data reads and writes. By definition:

- An ideal single data read takes 3 PCI clock cycles plus a turnaround cycle [4, p.133], but
- An ideal single data write takes 2 PCI clock cycles plus a turnaround cycle [4, p.141].

Now, this might indicate a small difference between reading and writing, but to further understand this difference, please consider the following analogies:

Writing

1. In a big meeting room, the boss yells that it is hot – and when the boss yells everyone becomes quiet – and that the air-conditioning must be turned on.
2. The employee near the air-conditioning commands listened (as everyone else did) and should now execute the boss's order.

Actually one may object that the employee can be distracted, thus making the boss repeatedly yell until it gets tired. In the PCI bus this is quite rare, for a reason similar to that eventually this employee will get a reprimand for not paying attention to its boss.

Reading

1. In the same office, the boss now yells for a coffee.
2. The employee responsible for coffees acknowledges this request, and goes to the coffee machine.
   (a) While the coffee is brewing, the boss is waiting, or
   (b) Alternatively the employee tells the boss the coffee will take some time, so he can pay attention to other things, asking for the coffee later.
3. Eventually the coffee is delivered to the boss.

Of course the reading analogy might look an intrinsically human process, but systems which have on-board memory have to cope with real latencies when accessing them. Just like both boss and employee have to wait for the coffee machine to warm up if it was off.

So, until the data is ready to be latched in the PCI bus, wait states are inserted by the target in bus by the target, effectively lengthening the transaction. Alternatively the target may just decide it's not ready to supply data when the master requests it and reply with a retry transaction, as described in [4, p.180-181].

2.2 Bus-mastering DMA

As we saw, reading has evident performance penalties. Thus, all modern add-on boards are designed so as to be masters and write their data directly to the system's main memory as a general rule. An exception to this rule are video frame-grabbers which write their data directly to the video adapter's memory.

This behavior closely follows the functionality provided by the now deprecated direct memory access controller, DMAC. This chip was used for boards in the ISA bus, and its derivatives like EISA. It was a device able to transfer data from a board directly into main memory, thus freeing the CPU to do something else.

Otherwise, CPU reading is a two-step process, performed by:

1. Reading data from the PCI bus to a CPU register, and then
2. Writing it from that register to the main memory.

Thus all modern add-on boards needing fast access to main memory are bus-masters and perform bus-mastering DMA. The most common examples of these are network interfaces and hard drive controllers.

2.3 The FLIC

The main motivation for this work waste decision of not implementing standard bus-mastering capabilities to the FLIC, although the ORCA FPGA in the FLIC allows for it.

This choice was made because:

1. There was a tight schedule for other FPGA software development like the FLIC's internal SDRAM controller or PCI glue logic;
2. Because managing the board inside the operating system becomes more complex since interrupt handling implies the compulsory development of a device driver with an appropriate interrupt service routine – ISR – and
3. Because proton runs do not need PCI's maximum data throughput given the amount of data generated.

3 PCI reading with the CPU

Since the ideal single data read transaction takes at least 4 PCI clock cycles, one has a theoretical maximum throughput of $\sim$31.5 MB/s on a 32-bit, 33 MHz bus. This suffices for proton runs, hence efforts have been taken into understanding how to use the CPU to read PCI memory.

3.1 Computer system's architecture concepts

Before entering the details on how the CPU reads data from an agent on the PCI bus, it is useful to understand some points of modern computer architectures. This explanation aims to be effective rather than fundamental.

Computer systems have two important components: the north and south bridges – NB and SB. These are mainly responsible for CPU interfacing and I/O interfacing, respectively.

Modern computer's chipsets present two essentially different ways of using the PCI bus with respect to the NB and SB:

NB $\overset{\text{PCI}}{\Longleftrightarrow}$ SB

This is the classic approach. It can be found on Intel's i440BX old chipset (depicted in fig. 2 on the next page) and AMD's 760 recent chipset, fig. 3 on the facing page.

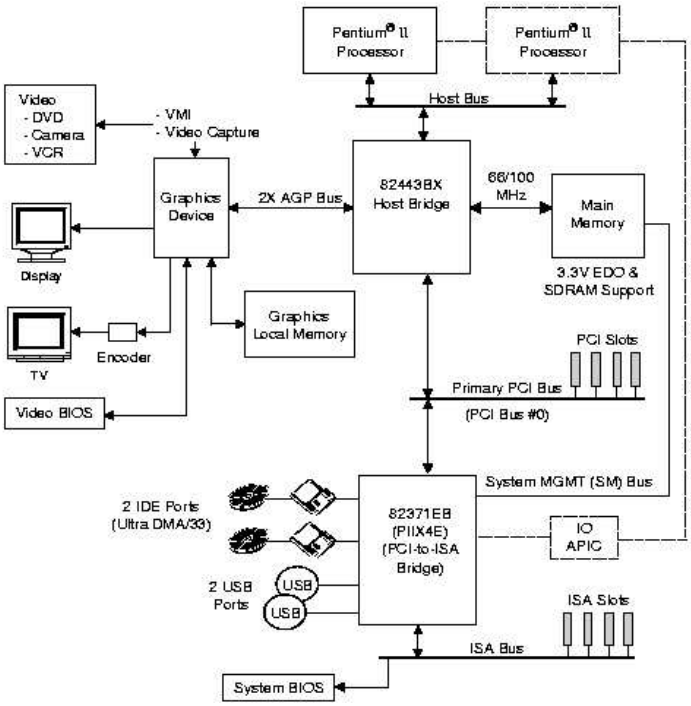


Figure 2: The Intel 440BX AGPset architecture.

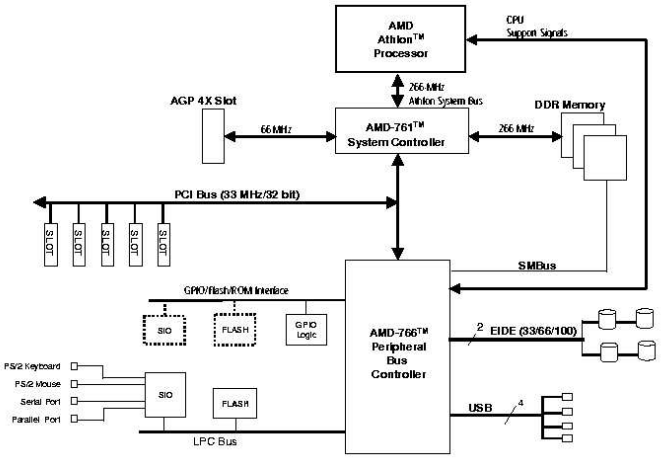


Figure 3: The AMD 760 chipset architecture.

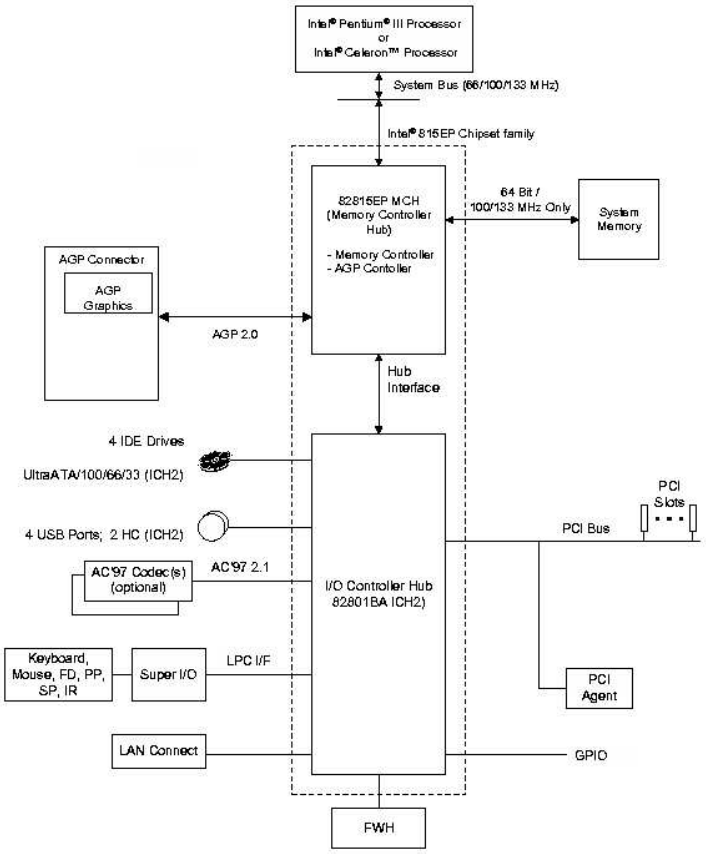


Figure 4: The Intel 815EP chipset architecture. Notice the PCI bus placed behind the ICH. Also notice most common peripherals directly integrated into the ICH.

These chipsets use the PCI bus to provide access to peripherals integrated in the PCI-to-ISA bridge. The PCI-to-ISA bridge appeared when systems migrated from ISA to PCI. When the ISA bus became completely deprecated, the approach was still maintained so as to have a standard PCI way of adding integrated peripheral controller chips. This way the north bridge *is* a PCI device able to communicate directly with other agents on the PCI bus.

Historically this approach is sensible, since when PCI came to overtake ISA, its bandwidth was as large as the memory to CPU bandwidth. Since then, things have changed and a different approach has arisen.

NB $\overset{x}{\Longleftrightarrow}$ SB $\overset{PCI}{\Longleftrightarrow}$

This is a quite recent approach. It can be found on Intel's i815EP chipset (fig. 4) or VIA's Apollo Pro266 chipset, fig. 5 on the facing page.
These chipsets use a proprietary bus linking the north and south bridges, named by Intel Memory Controller Hub (MCH) and I/O Controller Hub, ICH, respectively. This way the PCI bus is always behind the SB.

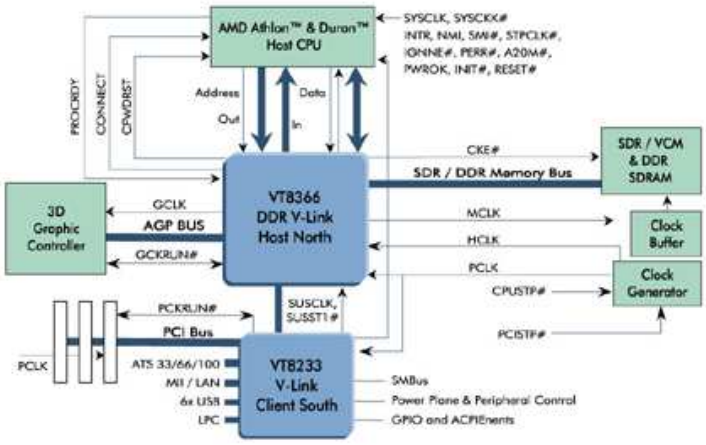


Figure 5: The VIA Apollo Pro266 chipset architecture. Notice the V-Link bus connecting the north bridge (V-Link host) to the south bridge (V-Link client).

This way of placing the PCI bus implies that integrated peripherals do not use PCI resources. This might significantly reduce traffic on the PCI bus, since ATA100 drives can continuously lurk up to 40 MB/s. *E.g.*, VIA's V-Link (fig. 5) has a total bandwidth of 266 MB/s. This effectively decouples the SB's integrated peripherals' needs from the PCI bus.

Since AMD focus on processor development, it is believed that they kept the old approach to allow board manufacturers to combine the AMD-761 PCI-compatible NB with a SB from the several in the market offering integrated peripherals like ATA100, networking, USB and other.

The most recent and first multi-processor (MP) chipset from AMD – the AMD 760MPX – presents the interesting feature of using a 64-bit, 66 MHz PCI bus as the interconnect bus, as can be seen in fig. 6 on the following page. This kind of interconnect has the impressive capability of handling up to 504 MB/s, while being PCI compatible.

Chipset choice has revealed to be a crucial factor in CPU initiated reads, since this is the CPU's frontend to the PCI bus.

3.2 Linux and PCI - operating system's architecture concepts

This discussion will be limited to Linux 2.2.x, NA60's DAQ operating system – henceforth OS always refers to Linux 2.2.x. Reasons for this choice include compatibility with other software, namely ALICE's DATE, among others.

From Linux the PCI bus is not directly accessible, because each CPU architecture implements it's own virtual memory addressing scheme. Since Linux aims at being portable between several architectures, there is a common interface entailing three different types of address spaces:

Real memory Mapped to the physical RAM on the computer.

System I/O Mapped to the I/O ports of devices present on the system.

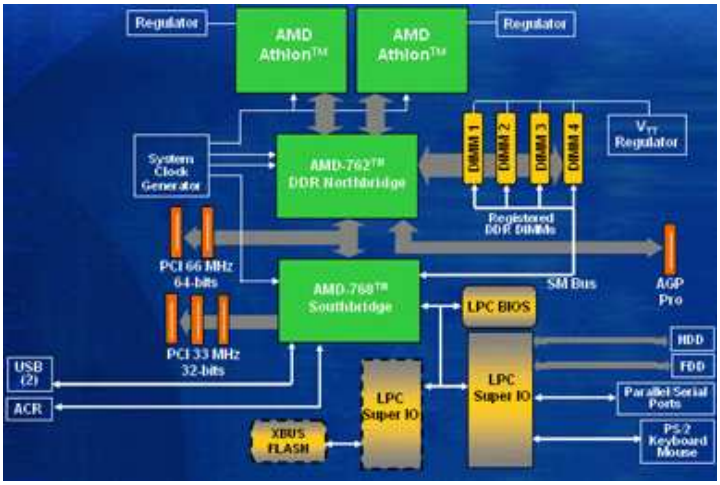


Figure 6: The AMD 760MPX chipset architecture. Notice the 64-bit, 66 MHz PCI bus used between NB and SB.

I/O memory Mapped to the memory of I/O devices like PCI add-on boards.

In the x86 architecture, real memory and I/O memory are on the same address space, such that directly dereferencing pointers on these spaces works. This is not portable, and is considered bad practice. Quoting [5, Documentation/IO-mapping.txt]:

> You should never take the address of such [PCI] memory, because there is really nothing you can do with such an address: it's not conceptually in the same memory space as "real memory" at all, so you cannot just dereference a pointer. (Sadly, on x86 it _is_ in the same memory space, so on x86 it actually works to just deference a pointer, but it's not portable). [...] You may think that all the above is overly complex, but one day you might find yourself with a 500 MHz Alpha in front of you, and then you'll be happy that your driver works ;)

3.2.1 Accessing PCI from Linux

Accessing PCI from Linux follows similar path for both kernel- and user-mode programs. Two steps are always needed:

1. Locating the device being accessed and determining the resource to be used, and then
2. Getting a valid handle to manipulate the resource from the OS.

Code for running in kernel-mode, dubbed a kernel module, can be easily developed as a normal C program which is dynamically linked to the kernel at run-time with the `insmod` utility. Kernel modules then become effectively a part of the kernel, and have special privileges, which even the superuser (`root`) in user-mode hasn't. These include executing some assembly instructions, as we'll see later.

Host to PCI Bridge (Device 0, Function 0)				Offset
Device ID		Vendor ID		0x00-0x03
Status		Command		0x04-0x07
Class Code = 0x060000			Revision ID	0x08-0x0B
Reserved	Header Type	Latency Timer	Reserved	0x0C-0x0F
BAR0 - AGP Virtual Address Space				0x10-0x13
BAR1 - GART Memory-Mapped Control Registers Pointer				0x14-0x17
Reserved				0x18-0x1B
Reserved				0x1C-0x33
Reserved			Capabilities Pointer: A0	0x34-0x37
Reserved				0x38-0x43
Extended BIU Control				0x44-0x53
ECC Mode/Status				0x48-0x4B
PCI Control				0x4C-0x4F
AMD Athlon™ System Bus Dynamic Compensation				0x50-0x53
DRAM Timing				0x54-0x57
DRAM Mode/Status				0x58-0x5B
Reserved				0x5C-0x5F
BIU0 Status/Control				0x60-0x63
BIU0 SIP				0x64-0x67

Figure 7: AMD-761 north bridge configuration space. The shaded area represent the PCI standard defined configuration space.

3.2.2 PCI Configuration Space

PCI configuration space is a 256-byte long (64-dword) area associated with each PCI agent, described in [4, ch.17]. In this space all the agent's PCI configuration, status, capabilities and resources are described. Figure 7 shows the configuration space for the host bridge PCI agent on the AMD-761 north bridge PCI device.

In user-mode, the main tool used to interact with the PCI configuration space of an agent is the `pciutils` package, [6]. This package provides the `lspci` and `setpci` utilities as well as the `libpci` library.

`lspci` allows direct checking of PCI configuration spaces. A sample output of `lspci -tvv` on an i440BX-based system, follows:

```
[root@pcru00 /root]$ lspci -tvv
-[00]-+-00.0  Intel Corporation 440BX - 82443BX Host bridge
      +-01.0-[01]-00.0  ATI Technologies Inc 3D Rage Pro AGP
      +-04.0  Intel Corporation 82371AB PIIX4 ISA
```

```
+-04.1  Intel Corporation 82371AB PIIX4 IDE
+-04.2  Intel Corporation 82371AB PIIX4 USB
+-04.3  Intel Corporation 82371AB PIIX4 ACPI
+-06.0  Adaptec AHA-2940U2/W / 7890
+-09.0  3Com Corporation 3c905B 100BaseTX [Cyclone]
\-0b.0  Ensoniq ES1370 [AudioPCI]
```

This is a tree like processor-centric view of the bus. The descriptive text appearing next to each agent is derived from a database linking the agent's vendor and device IDs (VID, DID) to an appropriate string describing the agent.

Note the way an agent on the PCI bus is addressed; this is accomplished by designating the bus number, device number and function number for that agent. Thus the IDE controller agent on the system above, is function 1 of device 4 on bus 0, or `0:4.1`.

Also note that the AGP bus is seen as PCI bus 1 with device `0:1.0` acting as the PCI-to-PCI bridge between PCI bus 0 and the AGP bus, emulated as PCI bus 1. In fact, this is only done in order to get a consistent picture of the whole system, because transactions between PCI and AGP buses are severely restricted, *cf.* [11, p.108].

Conversely to `lspci`, `setpci` allows you to change parameters of an agent's configuration space.

`libpci` is a library providing all of the functionality from `lspci` and `setpci` to compiled code.

All these are user-mode tools, and, *e.g.* `setpci` has to be run with `root` privileges; Pandora's box would crack open if any user could freely fiddle with the PCI sub-system configuration...

When working from inside the kernel, equivalents to all the functions in `libpci` are naturally available, since it is the kernel's task to manage system's peripherals.

3.2.3 PCI resource assignment

Memory and I/O resource addresses are assigned by the BIOS to the boards at boot time before the OS is loaded. Roughly speaking,

1. The BIOS sequentially repeats the following actions for all agents in the bus:
   (a) It writes 0xFFFFFF (32-bit "all ones" value) into each base address register (BAR) of the agent.
   (b) It reads back the value on that BAR. This value determines the resource type – I/O or memory – and in the memory case, its length.
2. The BIOS then determines a non-conflicting distribution of addresses – and ranges in the memory case, fitting all agents' requests.
3. Each BAR is then written with the address determined in the previous step.
   This way the PCI agent knows that when this address is requested on the bus, it should act upon the request.
4. The OS is then loaded and this information is stored into an internal data structure. A Plug-and-Play OS might still reconfigure some things if needed. The final system configuration is available inside the kernel for direct manipulation.

```
/* Here we assume that BAR0 (offset 0x10) is of memory type
 * Then pcilib's pci_read_long function is used.
 * Then the value returned is appropriately masked.
 */
pci_mem_addr = pci_read_long( pci_device_config_space, 0x10)
    & PCI_BASE_ADDRESS_MEM_MASK;

/* We now open the memory special device
 */
fd = open ( "/dev/mem", O_RDWR );

/* And finally import the PCI memory area to our process
 * (man 2 mmap for more info)
 */
ptr_to_pci_mem = (char *) mmap(
    NULL, PCI_MEM_LEN,
    PROT_READ|PROT_WRITE, MAP_SHARED, fd, (off_t)pci_mem_addr
);
```

Table 1: User-mode code to access PCI memory.

3.2.4 PCI I/O space

As far as I/O goes, both in kernel- and user-mode – with `root` privileges implied – one uses the `{in,out}{b,w,l}()` family of functions (input and output of byte, word and long data to an I/O port) on the I/O address read from a BAR declaring an I/O resource.

It is well known that I/O programming is *very* slow for data transfer, so I/O was not studied and will not be used in NA60's implementation of the FLIC, as can be deduced from [3]. For the NA60 FLIC, all I/O operations will be done via memory-mapped registers.

3.2.5 PCI Memory space

User-mode access

The first breakthrough on PCI memory reading was discovering how simple in Linux it is to do so. A user with `root` privileges, writing a C program just needs:

1. to `open()` Linux's special device `/dev/mem`, and then
2. use the `mmap()` system call to map the PCI memory into its process's memory space.

The code in table 1 illustrates this.

Kernel-mode access

From inside the kernel, everything is simpler, since information about PCI agents on the bus has been setup at boot time and is available for manipulation. In the kernel you still need to get a valid pointer to the PCI memory. So you use the analogue of `mmap()`, `ioremap()` with the PCI address obtained from the kernel's internal PCI structures as its argument. Code illustrating this can be found in table 2 on the next page.

```
/* Get a pointer to the device with the given VID and DID
 */
pcidev = pci_find_device(VENDOR_ID, DEVICE_ID, pcidev);

/* Retrieve the value of BAR0 to get the PCI address of the
 * memory area
 */
pci_mem_addr = pcidev->base_address[0] & PCI_BASE_ADDRESS_MEM_MASK;

/* Finally map the area into kernel space
 */
ptr_to_pci_mem = ioremap(pci_mem_addr, PCI_MEM_LEN);
```

Table 2: Kernel-mode code to access PCI memory.

PCI memory manipulation

Both user- and kernel-mode methods provide you with a valid pointer that links the current process's memory space to the PCI memory.

So, as expected, you can now use any standard functions from `string.h`, like `memcpy()`, `memset()` and `memmove()`. This leads us to the first tests of PCI reading.

4 PCI reading tests

This is a chronological description of the tests that lead to the actual state of knowledge on the subject.

Throughout this section, read speed increases are bold-faced, and any significant events are used to subsection the section's flow.

4.1 ATI VGA on PCI and AGP

Testing started on both PCI and AGP versions of the ATI 3D RAGE PRO graphics adapter. The first program flipped every bit in the adapter's memory, effectively turning an X-Windows screen into it's negative.

First results on the AGP board pointed to 8.5 MB/s read speed on an i440BX system. This system is a dual Pentium II at 400 MHz on an Asus P2B-S motherboard.

The speed was measured by running a user-mode program based on `/dev/mem` `mmap`'ing and `memcpy`'ing, as discussed earlier. Timing was performed using Linux's `time` utility. The tests have been performed in single-user mode, achievable issuing `init 1` as `root`. In this mode every service in the system is shutdown and the user is presented a superuser shell.

When the same tests where repeated using the ATI PCI, the read speed dropped to around **6.9 MB/s**, spanning from 6.2 to 7.2 MB/s depending on the transfer size. This drop is natural, since the AGP bus besides having a point-to-point topology, also has a higher clock speed.

The system was then taken to runlevel `5(init 5,)` which corresponds to networking and X-Windows running – and the read speed value dropped ∼9%.

4.2 Dolphin's PCI-SCI board

Next, a Dolphin PCI-SCI board, having 3 PCI memory areas with 32kB each was tested. Its comparison to the ATI PCI follows:

Board read speed (MB/s)	Dolphin	ATI PCI
single-user mode	8.4	6.9
runlevel 5	7.6	6.3

The ratios between speeds in each runlevel are almost the same, so system load seems to affect card reading in the same way.

4.3 Writing to the PCI memory

Afterwards, read performance was compared to write performance. For the ATI PCI board in single-user mode, we got 40 MB/s writing with the same 6.3 MB/s reading in runlevel 5. At this time the writing *vs.* reading issues were not known, so these differences presented the intriguing question of why writing was *so* fast.

Since the Dolphin board tended to hang up the system when written, and no documentation on how to use its memory areas was available, it was withdrawn from further testing, although it looked 20% faster than the ATI PCI.

A SCSI controller was then used to do some reading/writing tests, giving the disappointing read and write values of

Board/Device	SCSI	ATI PCI	ATI AGP
Reading (MB/s)	2.2	-	-
Writing (MB/s)	12.6	31.4	29.5

4.4 Using a kernel module

After these user-mode programs, a minimal device driver was written. This used the family of `memcpy_{to,from}io()` functions to move the data to/from the PCI `ioremap`'ed memory. To time the transfers, this driver used the kernel variable `jiffies`, which in the x86 architecture is incremented every 10 ms. Extensive information on these functions and the Linux kernel's internals can be found in [7].

This driver was used to read and write blocks of 64 kB data. In runlevel 2 – networking multi-user mode without NFS – the results were:

	ATI PCI – 64kB blocks
Reading (MB/s)	**9.1**
Writing (MB/s)	103.2

So, suddenly writing became three times faster at 82% of PCI maximum bandwidth, while reading "only" increased ∼30%.

Further testing using blocks of 1 and 2 kB, in both runlevels 3 and 5 gave 8.6 MB/s reading and 102.4 MB/s writing.

4.5 3Com's 3c905B NIC

Next step was considering a network interface controller, NIC, since these are highly developed and proven boards. A 3Com 3c905B was tested. This board bears an 8 kB Rx/Tx buffer.

In runlevel 1, with `syslogd` running so as to be able to trace the driver's messages, and using blocks of 1 kB, the driver measured

	3c905B NIC – 1kB blocks
Reading (MB/s)	8.4
Writing (MB/s)	16

So one can see that both the SCSI and the 3c905B devices perform very poorly on expectations. From our experience this just means they're optimized to do themselves the job, in such a way that when the CPU tries to do it for them, they simply misperform.

4.6 Tweaking in the kernel

At this time, the Linux kernel sources were searched for what the `memcpy_{to,from}io()` functions were doing. In fact these are just address translating wrap-ups to the more fundamental `__memcpy()` function. In turn this is an assembly coded function using 32-bit MOV assembly instructions to transfer the data in tight loop.

While tracing these functions, other `memcpy`-type functions were found. One had the promising name of `best_memcpy()`, and the other was `__constant_memcpy()`.

Also kernel memory allocation flags in `kmalloc()` were varied, yielding no significant read speed differences. The maximum memory allocatable in the kernel is 128 kB, since it is real rather than virtual memory.

Results for the ATI PCI board gave

ATI PCI read/write speed (MB/s)				
Block datasize	32 bytes	128 kB		
`kmalloc()` flags	`GFP_KERNEL` (default setting)		`__GFP_HIGH | __GFP_DMA`	`__GFP_HIGH`
`memcpy_from_io()`	7.83; 336.84	9.08; 103.22	8.99; 103.22	9.09; 103.22
`__memcpy()`	7.83; 336.84	9.09; 103.22	9.12; 101.58	9.12; 101.58
`__constant_memcpy()`	7.83; 376.47	9.13; 103.22	9.14; 103.22	9.14; 103.22
`best_memcpy()`	7.76; 200.00	9.16; 103.22	9.16; 103.22	9.17; 103.22

These tests are rather inconclusive, although `best_memcpy()` looks just slightly better at reading larger blocks of data. The +300 MB/s measured speeds for writes are still an unexplainable fact, but this might be due to the fact of 32 byte writes being merged in some fashion. Anyway, writing does not present a relevant issue to this work.

Further investigation in the kernel sources about the `best_memcpy()` function revealed a wrap-up deciding between using `__memcpy()` and `kni_memcpy()` depending on the processor's support of streaming SIMD extensions – SSE. SIMD instructions are single-instruction multiple data instructions, introduced in the MMX enabled CPUs. The SSE instructions were introduced in the Pentium III revision of Intel's IA-32 x86 processors. Testing was being performed in a PII processor, so `kni_memcpy()` was never chosen by `best_memcpy()` to do the final assembly work.

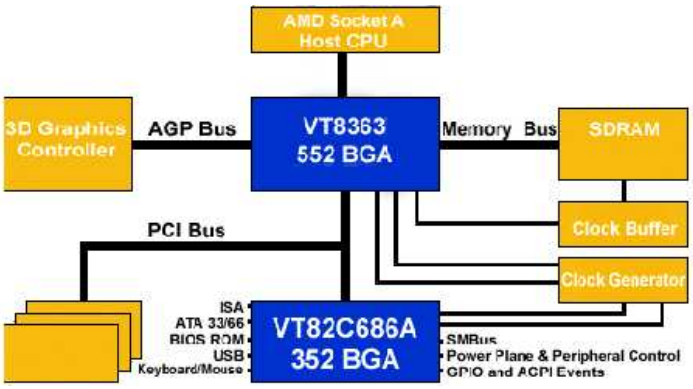


Figure 8: The VIA KT133 chipset architecture.

4.7 The VIAKT133

At this point testing was done in an Athlon based computer with an Asus motherboard with a VIAKT133 chipset. This chipset has the classical PCI bus positioning and is depicted in fig. 8.

Using the ATI PCI and the user-mode program reading blocks of 4 MB, 7.46 MB/s where achieved. Using the kernel module one could read at the same speed 7.46 MB/s and write at 63.05 MB/s, no matter which `memcpy` routine was used.

Changing in the BIOS setup the video memory cacheability from US to USWC – uncacheable, speculative write combining – managed the write speed to go up to 119.6 MB/s. Anyway, changing other NB related parameters did not alter the situation. By combining smaller data writes into 64-bit writes, USWC reduces the number of transactions required for a particular amount of data to be transferred into the linear framebuffer of the graphics adapter. The factor of two observed corroborates this line of thought.

Anyway it was strange that in this case moving from user-mode to kernel-mode did not yield any speed increase. The reasons for this behavior will become clear latter on.

4.8 The i440FX

So as to get a third “chipset opinion”, a motherboard based on Intel’s i440FX was used on a system bearing a Pentium II at 233 MHz. This chipset is the uniprocessor version of the i440BX, already discussed. Results are summarized below:

	ATI PCI – 128 kB blocks
Reading (MB/s)	7.0
Writing (MB/s)	36.2

One can see that the performance is comparable to all the other systems tested up to this time: the i440BX and the VIA KT133.

4.9 Loop variable declaration

It has also been tested the way the variable that loops the block transfer is declared:

Loop variable type	ATI PCI – 128 kB blocks read speed (MB/s)
`u32`	9.1
`register u32`	**9.8**

When a variable is declared with the `register` keyword, the compiler makes sure this variable is kept in a register for as long as possible during execution of a program. The increase in speed though, is barely 10%. `u32` stands for a 32-bit unsigned integer type.

4.10 The manufacturers

At this time email was sent to major chipset/motherboard manufacturers like AMD, VIA, Opti, Ali and Serverworks. Intel and SiS had no technical support email contact available. This email simply asked if their NBs/motherboards supported CPU initiated PCI burst reads, since 9 MB/s would mean, in a 32-bit, 33 MHz PCI bus one data every 14 clock cycles.

4.11 Moving up to Pentium III

Meanwhile, and so as to test `kni_memcpy()`, a Pentium III at 800 MHz computer, based on the Intel's i815EP chipset, was ordered. The first results for this platform with the kernel module (runlevel 1 with `syslogd` implied) were:

ATI PCI – 128 kB blocks read/write speed (MB/s)	

`memcpy_from_io()`
`__memcpy()`
`__constant_memcpy()`
	5.77 / 64.00

`best_memcpy()`	8.95 / 112.28
pointer dereferencing	1.50 / -

So, `best_memcpy()` is now choosing `kni_memcpy()` because we're using a Pentium III, and both reads and writes get boosted by some 60%.

Pointer dereferencing is included, because it's similar to I/O access, by taking just on dword at a time. This technique just dereferences the pointer returned by `ioremap()`:

```
(unsigned long) value = *(unsigned int *) ptr_to_pci_mem;
```

With the user-mode program `mmap`'ing the PCI memory on `/dev/mem`, one got for 8 MB blocks the small read speed of 5.61 MB/s.

Coming back to the driver, but now using smaller data blocks one measured

ATI PCI – 1 kB blocks read/write speed (MB/s)	

`memcpy_from_io()`
`__memcpy()`
`__const_memcpy()`
	5.77 / 64.00

`best_memcpy()`	9.14 / 117.43
pointer dereferencing	1.50 / -

And tweaking the data block size one could see that `best_memcpy()` produced the best results in all runs from 512 bytes data per block up to 32 kB data per block.

When a 64 bytes per data block run was performed, then all the routines performed equally bad:

ATI PCI – 64 byte blocks read/write speed (MB/s)	

`memcpy_from_io()`
`__memcpy()`
`__const_memcpy()`
	5.77 / 64.00

`best_memcpy()`	5.77 / 62.13
pointer dereferencing	1.50 / -

This happens because for transfers smaller than 128 bytes, `kni_memcpy()` internally prefers to use `__memcpy()`.

4.12 Using the PCI Analyzer

At this time, the TA700 PCI analyzer/exerciser from Catalyst Enterprises, was brought from the lab in order to ascertain the differences seen in terms of PCI cycles. The main motivating factor was the observed two fold increase in speed either reading and writing, when `kni_memcpy()` was used.

The analyzer showed that `kni_memcpy()` retrieved 2 dwords per read transaction, while all the other routines only got one dword per read transaction. Now, an XMM register used by SSE instructions on `kni_memcpy()` is 128-bit wide, and 2 dwords fit only half of it. So, this hints at a bottleneck/mismatch somewhere. Otherwise we'd expect to get 4 dwords (128 bits) bursted in each read transaction.

4.13 Overclocking

At this point, another approach at increasing the read speed was taken: overclocking the PC.

Since the FLIC is a 64-bit 66 MHz PCI compliant board, overclocking would never be limited by the FLIC, but by specifications or quality of other system's components.

It was observed that overclocking the PCI bus to 41.77 MHz increased read speed of the ATI PCI to **10.63 MB/s**, or 16% more than the previous highest value.

Overclocking is very tricky, since the whole system is overclocked, including the system's front side bus (FSB) and the main memory speed. The CPU internal speed is determined by FSB times a fixed multiplier, so the CPU itself will also be overclocked.

Since the ATI PCI refused to let the system boot above 41.77 MHz, testing was redirected to the system's NIC, now an Intel EtherExpress PRO, which worked with a PCI bus frequency of 43.33 MHz, with FSB at 130 MHz and RAM at 130 MHz. Top read speed of 13.34 MB/s for this NIC was attained at PCI bus speed of 42.53 MHz, with FSB at 170 MHz and RAM at 128 MHz, and transferring 4 kB blocks. At this point the CPU was running at 1020 MHz. Notice that memory clock is now below the 133 MHz specification for PC133 SDRAM. Thus memory performance was degraded. Underclocked memory is the kind of compromise necessary in overclocking other components, because memory chips are usually the most clocking sensitive devices on a system.

With the system overclocked to 160/160/40 MHz on FSB/RAM/PCI, several tests where undertook on the EEPRO, using a private copy of the `kni_memcpy()` function. No performance improvement was observed by changing the code, and these changes included:

- Interleaved reading of several offsets;
- Changing the location of PREFETCHNTA instructions;
- Adding more PREFETCHNTA instructions;
- Removing all PREFETCHNTA instructions;
- Unroll parts of the copy loop using more XMM registers;

As we now know, the PREFETCHNTA instruction, defined in [13, p.10-18] doesn't necessarily prefetch a cache line, namely due to issues related with cacheability of the address being asked for.

When the analyzer was brought in, the system would refuse to boot, so no further testing was performed in an overclocked system.

4.14 The TTC/SR PMC

At this time the first assembled FLICs arrived to the lab. The only active PCI device similar to a FLIC available for testing was the TTC/SR PCI mezzanine card – PMC – which has its own on-board PCI-to-PCI bridge. Adding this bridge to the one on the FLIC, this means the TTC is now an agent behind two bridges, as seen by the host system. This has revealed to be an issue, since the bridges forward PCI requests to the TTC, thus introducing delays when the CPU reads it.

The first time the analyzer was used to snoop on TTC reads, a brave new world was uncovered.

The notation [x+y] used for read transfers means that offsets x and y are transferred in the same PCI read transaction. These offsets are always an hexadecimal multiple of 0x04 because 4 bytes (1 dword) are transferred at a time in the PCI bus. Several instructions were used to transfer data:

- MOVUPS instructions on XMM registers (128-bit wide) interlace the data on the bus. By this we mean that the order by which data in PCI memory are read is: [08+0C] [00+04] [18+1C] [10+14]...

  MOVUPS are the instructions used by `kni_memcpy()`.

- MOVDQ operations on MMX registers (64-bit wide) do not interlace the data.
  For these one can see [00+04] [08+0C] [10+14] [18+1C]... on the bus.
- MOV operations on 32-bit registers result in the expected pattern of [00] [04] [08] [0C] [10] [14] [18] [1C]...

In fact each [x] bracket is effectively a group of PCI commands, which can be divided in two sets:

1. Three transactions consisting of

   ```
   3x (MEMRD + WAIT + RETRY + WAIT + 2 IDLE),
   ```

   amounting to 18 PCI clocks where no data is transferred, and
2. A final data transferring transaction consisting of

   ```
   (MEMRD + 2 WAIT + {1 or 2} DATA + ~22 IDLE),
   ```

   amounting to ~27 PCI clocks, with 1 or 2 dwords transferred depending on the function used.

This clearly explains the big speed difference between `kni_memcpy()` and all the other functions. Briefly stated one has

Function used	Average bytes transferred per PCI clock cycle
`kni_memcpy()`	0.178
other `memcpy`'s	0.091

From this, one had two presumably exploitable ways of increasing the bytes/clock ratio:

1. Understand why there are three retries but looking at the PCI transactions in between the TTC and the FLIC, rather than outside the FLIC.

   This proved to be physically impossible, since if we use the analyzer plugged in the FLIC, then we couldn't plug the TTC there.

   Anyway this method of diagnosis may be user for checking the FLIC, since then one will see the transactions between the FLIC's on-board bridge and the FPGA itself.
2. Try to reduce the IDLE states between consecutive transactions. Halving the IDLE states would increase the speed by 32%.

   An explanation for the amount of IDLE states conjectured is that the NB after getting the data, hands it to the CPU which, after filling a register, now has to write this register contents' to memory. This operation was taking roughly 87 clock cycles of the 133 MHz FSB, which is more than enough time to store 64 bits into memory. The IDLE states could then be reduced by using a faster CPU and/or faster memory. Thus a PC with an Athlon processor at 1 GHz equipped with double data rate, DDR, SDRAM was ordered.

The conclusion drawn from the results above was that it seemed impossible to get more than 2 dwords at a time from the PCI bus.

4.15 AMD's opinion on CPU initiated reads

At the same time this was being unveiled, AMD replied to the email asking for directions. They were the only manufacturer who replied to our message, and all these previous findings were shared with them.

The idea they were passing was that NBs are optimized for everything *except* burst reading from PCI boards, and that this *is* very natural. Quoting from [8, p.17], one has that

> The PCI block [the north bridge] can be broken up into two sub-blocks–the PCI target module and the PCI master module. The PCI target module handles cycles initiated by an external master on the PCI bus. The AMD-761 system controller responds to cycles that are directed to main memory or writes that are sent to the other PCI interface (the AGP interface). This module contains *write buffers (PCI-to-memory and PCI-to-PCI)*, *read buffers from memory*, and a target sequencer that keeps track of the bus while the AMD-761 system controller is a PCI target.

The emphasized text shows that there are no read buffers from PCI. So it really looked that no NB would do the task. The last email received from AMD Europe, quoted an AMD USA senior chipset designer, who wrote:

> I don't know of any chipset that does this because it kinda goes against the whole PCI paradigm which is to move data with DMA-not the CPU. Moving data with DMA is more efficient but I have run across a few customers in the past that resist this because they don't want to spend the money[...].
>
> [...]
>
> So the next option you could support is read-prefetching, where the PCI master would speculatively go out and "read-ahead" in spite of the fact that the processor's read was uncacheable. As I mentioned we did this in a previous chipset at my last company (Pentium based) and it was a total disaster because you can't prefetch in non-prefetchable memory that has side-effects (i.e. a status register that resets itself when read). We just couldn't get people to understand how and when to set the attributes correctly so we took this out.
>
> Bottom line though, I think that this is a feature you wouldn't typically see, want, or need in a PC application. It costs money to build and test and can cause problems when used incorrectly, so it's not ever supported.

The overwhelming strength of the words was so big that it looked like this endeavor would stop with a mere 9 MB/s top read speed.

4.16 The turning point – Tricking the NB with MTRRs

Memory type range registers (MTRRs) were *the* big step in read speed increase.

The use of MTRRs was suggested by someone on the PCI-SIG mailing list reporting that for testing purposes they were declaring a prefetchable PCI memory as write-back, WB, on the CPU's MTRRs, [9]. For the IA-32 architecture, one has the memory types of table 3 on the next page, abridged from [13, sec.9.3].

Memory type	Cacheable	Writeback Cacheable	Allow Speculative Reads	Memory Ordering Model
Strong Uncacheable (UC)	No	No	No	Strong Ordering
Write Combining (WC)	No	No	Yes	Weak ordering
Write Through (WT)	Yes	No	Yes	Speculative processor ordering
Write Back (WB)	Yes	Yes	Yes	Speculative processor ordering
Write Protected (WP)	Yes for reads No for writes	No	Yes	Speculative processor ordering

Table 3: IA-32 memory types declarable in a MTRR.

We have verified that it suffices to declare the memory area as write through, because the relevant parameter is the cacheable attribute. Programming the MTRRs can be done from a command prompt using the Kernel `/proc` interface. Shell commands illustrating this are given in Table 4 on page 30.

In principle this is only effective if one has an agent with prefetchable PCI memory being declared in the MTRRs. Otherwise it might induce violations to the PCI specifications in some way, because non-prefetchable areas are evidently not cacheable.

So, first results using this trick gave some interesting results

TTC read speed with MTRRs (MB/s)						
Transfer method	No-MTRR	WC	WT	WB		WP
32-bit MOVs	3.319	6.65	*1280*	1750	*1600*	*1280*
64-bit MOVQs	6.72	-	-	2497	-	-
128-bit MOVUPSs	6.72	6.70	*711*	1883	*711*	*800*

Results in italics were obtained at the edge of the timing measurements precision. The `jiffies` variable precision is 10 ms and some measurements took only 40 ms. The results missing the 64-bit MOVQs were obtained with the usual `memcpy()` (32-bit) and `kni_memcpy()` (128-bit) functions with 64 kB blocks while columns having all three results where tight assembly loops transferring 32 bytes per iteration with the respective assembly instructions.

One can distinctively notice a transition when going to memory types allowing caching. One can also observe from the first WB column, that 64 bits looks like a preferred mode for transfer.

The huge values for WT, WB and WP MTRRs are an expected phenomenon, since a read from a cacheable memory area address makes the processor look for that address in its cache. Quoting from [13, p.9-6], for cacheable types one has that

> Reads come from cache lines on cache hits; read misses cause cache fills.

Thus, once a data is read into the cache, if this is not evicted, the probability of staying there during the repetitions of the transfers is rather high. The only thing that has effectively changed

inside the host bus is that a cache fill request is being issued, rather than a data request. This will prove to make all the difference.

Taking this into account, cache flushing becomes necessary at the end of each block transfer. If one doesn't want to flush the cache then the block size must be larger than twice the cache's size to guarantee that the whole cache is overwritten when the next block is read.

To prove this last point, imagine you have a L bytes cache and you are repeatedly reading a memory area of length l bytes. Then for the three possible cases outlined before, one has that:

$l < L$ When you first read your block, the whole l bytes are copied into the cache. Then, when you start reading that memory area again you will get a cache hit for the first byte, and all others subsequently.

$L < l < 2L$ For this case suppose $l = 1.5L$. Then the first time the memory area is read, only the last L bytes of the memory area are cached. So, when you read again, the first $L/2$ bytes are assured not to be in the cache, and will be fetched. But if due to cache organization you cannot guarantee that these $L/2$ bytes will overwrite the next $L/2$ bytes, then you need to make sure that $l > 2L$.
If one is sure that the cache is overwritten in a linear fashion, then one just needs to transfer $l > L$ blocks. But it is always safer to be prepared for the worst case.

4.17 Flushing the cache

Cache flushing has been achieved by coding the WBINVD assembly instruction in the kernel module. Quoting [12, p.3-780], the WBINVD instruction

> Writes back all modified cache lines in the processors internal cache to main memory and invalidates (flushes) the internal caches. The instruction then issues a special-function bus cycle that directs external caches to also write back modified data and another bus cycle to indicate that the external caches should be invalidated.
>
> After executing this instruction, the processor does not wait for the external caches to complete their write-back and flushing operations before proceeding with instruction execution. It is the responsibility of hardware to respond to the cache write-back and flush signals.
>
> The WBINVD instruction is a privileged instruction. When the processor is running in protected mode, the CPL of a program or procedure must be 0 to execute this instruction. This instruction is also a serializing instruction (see Serializing Instructions in Chapter 8 of the IA-32 Intel Architecture Software Developers Manual, Volume 3).

The line of code used reads

```
__asm__ __volatile__ ("WBINVD":::"memory");
```

But note that this piece of C code can only be used from within the kernel. Even the superuser cannot run a program calling it, as tested.

Using/not-using the MTTR trick along with the WBINVD cache flushing for 64 kB blocks on the Pentium III system with the TTC board and the Intel EEPRO network adapter yielded for `best_memcpy()`, in runlevel 1 with `syslogd` running,

(MB/s)	No-MTRR	WT MTRR	WB MTRR
TTC	-	**21.19**	21.19
EEPRO	8.88	13.31	-

A whopping factor two more than the last best results.

4.18 VIA's deception

After these fantastic results that plainly doubled the read speed, the idea behind Athlon testing was that the Athlon's cache line size is twice the Pentium III's: 64 bytes *vs.* 32 bytes, or 16 dwords *vs.* 8 dwords.

The Athlon board used the aforementioned VIA KT133. For this board, results using the new driver were very disappointing:

(MB/s)	No-MTRR	WT MTRR	WB MTRR
TTC	3.46	7.62	7.62

The analyzer clearly shows that only 2 dwords are bursted.

But the really disappointing thing is that when the MTRR is disabled there is only one dword being transferred per PCI read transaction, no matter what routine is used.

4.19 Back to Pentium III

After the frustrated hopes on the VIAKT133-based system, detailed analysis of the PCI bus' contents transfers on the Pentium III i815EP-based system was undertook.

These showed that for the TTC using WT MTRR, with WBINVD kernel module and transferring 64kB blocks, one has 20.85 MB/s. For this figure the transactions on the bus followed a pattern consisting of

1. The first retries

   ```
   3x (MEMRD + 2 WAIT + RETRY + WAIT + IDLE),
   ```

   amounting to 18 PCI clocks, and

2. The final data transferring transaction consisting of

   ```
   (MEMRD + 2 WAIT + 8 DATA + 17 IDLE),
   ```

   amounting to 28 PCI clocks, for a total of 8 data per 47 clocks.

This structure can be seen on fig. 9 on the next page. The theoretical throughput of this pattern is 21.75 MB/s, and its bytes/clock ratio is 0.68.

Now a very interesting effect due to the introduction of the WBINVD instruction is that the structure of the transfer at the 64 kB block level reveals an amount of IDLE time of 8.2% of the

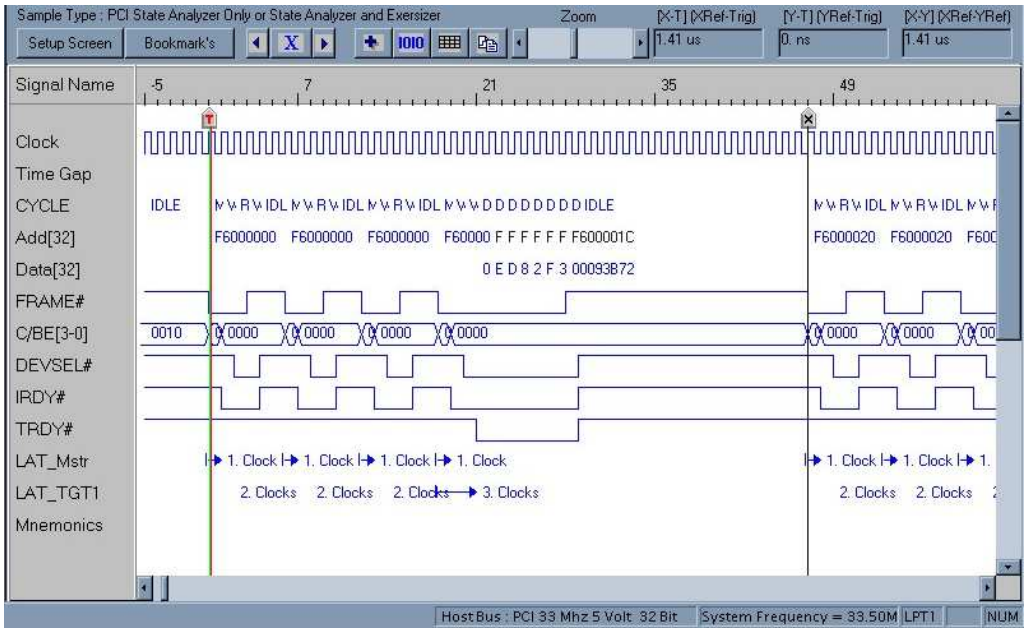


Figure 9: 8 dwords being bursted. Notice the timings on the upper right corner.

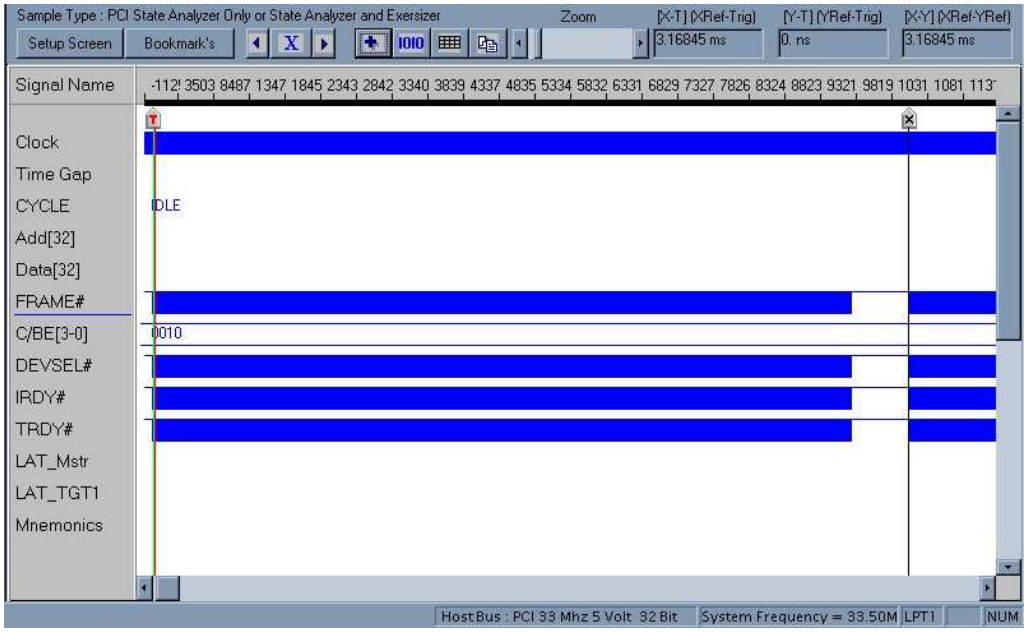


Figure 10: 64 kB transfer structure with 8 dword bursts. Notice the timings on the upper right corner.

total block transfer's time, inserted between block transfers.

This IDLE time is surely due to the cache flushing and block loop variable incrementing.

Since the TTC memory is 64 kB long, one could not go for the alternative to WBINVD, which is reading blocks twice as large as the cache size.

In the FLIC this will not be a problem since the FLIC's internal memory is 32 or 64 MB long. Thus when sequentially reading this buffer the cache will always be overwritten.

4.20 The AMD-761 System Controller

Before trying to get an expensive 128 byte cache line size Pentium 4 system to try getting more dwords per read transaction, the newly arrived AMD-761-based system with DDR SDRAM was tested. This has an Asus A7M266 motherboard fitted with 256 MB of DDR SDRAM.

First results using the same software used on the PIII system yielded

(MB/s)	No-MTRR	WT MTRR
TTC	4.1	**26.2**

This represents 23% faster reading by "just" changing CPU, memory and mainboard.

Looking at the bus one could see

1. Four first retries

   ```
   4x (MEMRD + WAIT + RETRY + WAIT + IDLE),
   ```

   amounting to 20 PCI clocks, and

2. The final data transferring transaction consisting of

   ```
   (MEMRD + 2 WAIT + 7 DATA + DIS+DATA + WAIT + 1 IDLE),
   ```

   or

   ```
   (MEMRD + 2 WAIT + 8 DATA + 9 IDLE),
   ```

   amounting to 13 or 20 PCI clocks, for a total of 8 data per 33 or 40 clocks.

The transfers observed in the bus had this intriguing alternating pattern of IDLE states after the data was transferred. This pattern alternated between 1 IDLE and 9 IDLEs. Further investigation revealed that:

- 9 IDLE states were observed when the transaction ended normally, with both master (NB) and target (the FLIC PCI-to-PCI bridge acting on behalf of the TTC) agreeing on termination.
- 1 IDLE state was observed when the target would disconnect before the master's needs were fulfilled.

The latter kind of disconnection is described in [4, p.180].

So, what seemed to be happening was that somehow, either the TTC or one of the PCI-to-PCI bridges were not giving all the data the NB was willing to receive. So the NB seemed to want 16 dwords, or an Athlon's whole cache line. This was very promising.

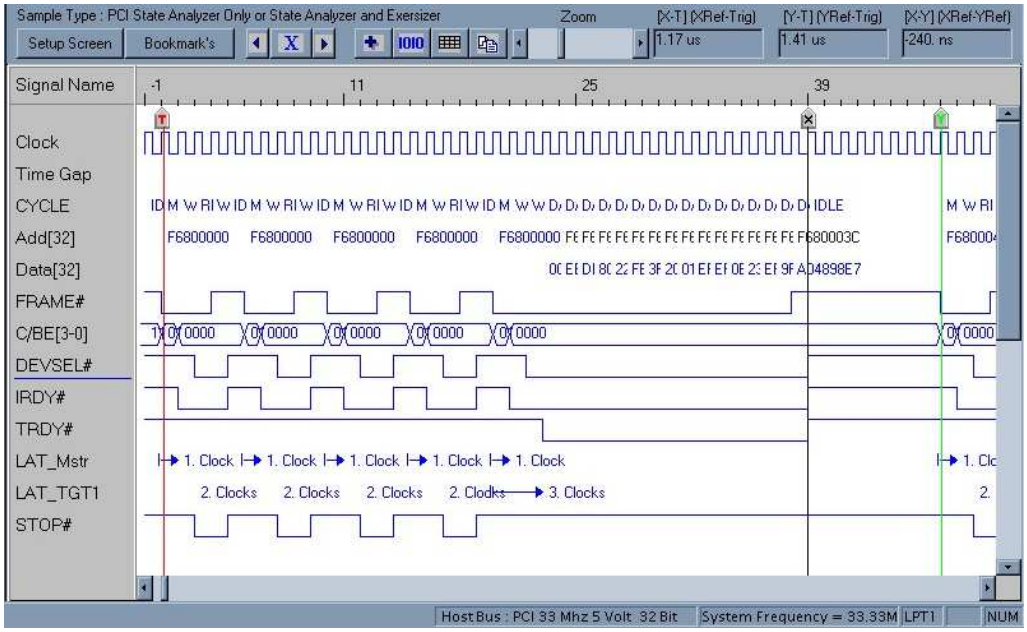


Figure 11: 16 dwords being bursted. Notice the timings on the upper right corner.

4.21 PCI-to-PCI cache line size tweaking

When investigated, the Intel 21154 PCI-to-PCI bridges present both in the FLIC and in the TTC board, revealed to be configured to a cache line size of 8 dwords. This value has a maximum setting of 16 dwords, so using the `setpci`, as shown in Table 4 on page 30 one changed this setting to the largest allowed value.

After this change one achieved

(MB/s)	No-MTRR	WT MTRR
TTC	-	**37.8**

In the bus one could observe the following transactions:

1. The four retries at the beginning

   ```
   4x (MEMRD + WAIT + RETRY + WAIT + IDLE),
   ```

   amounting to 20 PCI clocks, and

2. The final data transferring transaction consisting of

   ```
   (MEMRD + 2 WAIT + 16 DATA + WAIT + 11 IDLE),
   ```

   amounting to 30 PCI clocks, for a total of 16 data in 50 clocks.

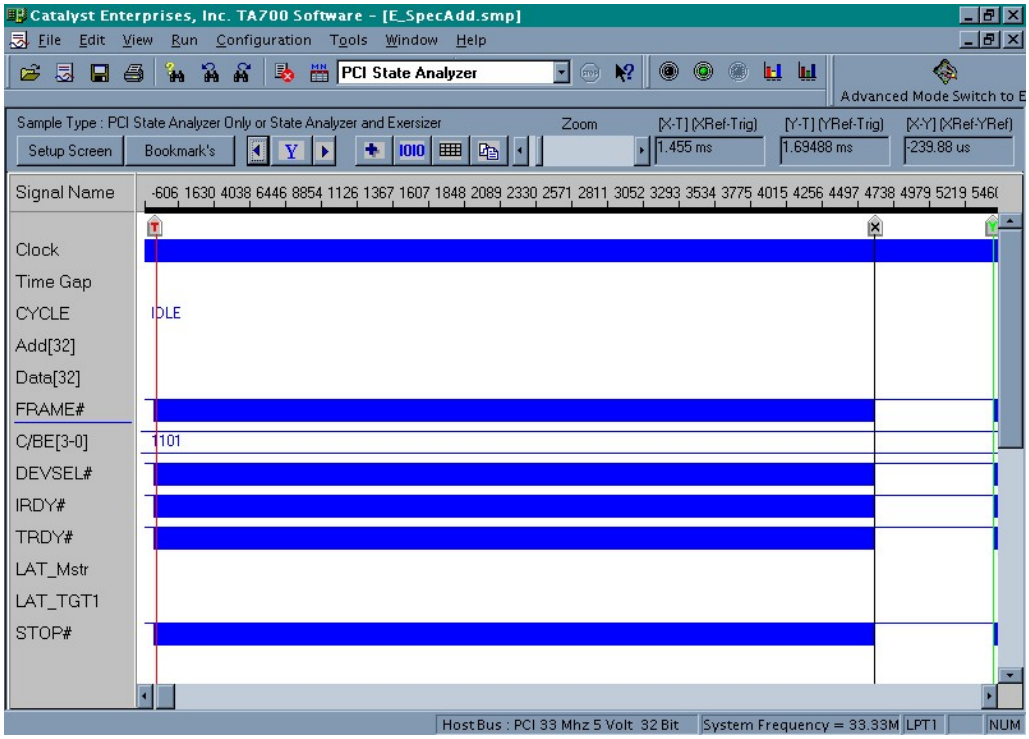


Figure 12: 64 kB transfer structure with 16 dword bursts. Notice the timings on the upper right corner.

These are depicted in fig. 11 on the preceding page. This pattern's theoretical throughput is 40.68 MB/s, and its bytes/clock ratio is 1.28.

The interesting bit is that now the inter-block delay has increased to 13.3% of the block's total transfer time, as can be observed in fig. 12. Further understanding of this time's nature would be very useful, but would probably need some peeking into the FSB. Since there are no FSB analyzers available this might prove impossible to do.

5 The 37.8 MB/s micro-howto

This section aims at providing a checklist on how to get 37.8 MB/s reading. We only mention key components for the reading purpose. In each component, relevant features are highlighted, and appropriate comments are drawn.

PCI Board We used a ***CERN PMC-TTC/SR*** board. The TTC/SR PMC was mounted in a FLIC – so that the FLIC is just a carrier. The TTC/SR declares a ***prefetchable PCI memory area*** which is 64 kB in length.

CPU The CPU used was an ***AMD Athlon*** at 1 GHz. This processor has a ***cache line width 16 dwords*** (64 bytes).

```
[root@na60pc13 /root]#lspci -tvv
-[00]-+-00.0  Advanced Micro Devices: Unknown device 700e
      +-01.0-[01]----00.0  ATI Technologies: Unknown device 474d
      +-04.0  VIA Technologies, Inc.: Unknown device 0686
      +-04.1  VIA Technologies, Inc. VT82C586 IDE [Apollo]
      +-04.2  VIA Technologies, Inc. VT82C586B USB
      +-04.3  VIA Technologies, Inc. VT82C586B USB
      +-04.4  VIA Technologies, Inc.: Unknown device 3057
      +-09.0  3Com Corporation: Unknown device 9200
      +-0d.0-[02-03]--+-0e.0-[03]----00.0  CERN TTC (PMC)
      |               \-0f.0  Lucent : FLIC board
      \-11.0  Promise Technology, Inc.: ATA100 controller
[root@na60pc13 /root]#lspci -vs 3:0.0
03:00.0 Bridge: CERN/ECP/EDU STAR/RD24 SCI-PCI (PMC) (rev 02)
        Flags: bus master, slow devsel, latency 32, IRQ 10
        Memory at e3000000 (32-bit, prefetchable)
        I/O ports at 8800
[root@na60pc13 /root]#echo "base=0xe3000000 size=0x10000
type=write-through" > /proc/mtrr
[root@na60pc13 /root]#cat /proc/mtrr
reg00:base=0x00000000(   0MB),size= 256MB:write-back,count=1
reg01:base=0xe3000000(3632MB),size=  64kB:write-through,count=1
reg07:base=0xe4000000(3648MB),size=  64MB:write-combining,count=1
[root@na60pc13 /root]#setpci -v -s 0:d.0 cache_line_size=10
[root@na60pc13 /root]#setpci -v -s 2:e.0 cache_line_size=10
```

Table 4: Tuning session with the AMD-761 system with the TTC/SR PMC. Keywords are highlighted.

Motherboard We used an Asus A7M266 motherboard. This motherboard used the ***AMD-761*** north bridge. Testing on another Athlon motherboard with the VIA KT133 chipset, keeping all other conditions constant, yielded only 2 dwords being transferred at a time.

Setup The system must be running Linux. Then as `root` perform the commands in Table 4. These set the MTRRs and PCI-to-PCI bridges cache line sizes' to 16 – 0x10 in hexadecimal. Besides, it is also mandatory that either:

1. The size of the memory area being repeatedly read is larger than twice the processor's total data cache size,

   or

2. WBINVD is issued at the end of every transfer. This can only be done from inside the kernel and using the `__asm__ __volatile__ (“WBINVD”:::“memory”)` construct.

6 Prospects

From all the facts observed, probably the achieved read speed is the highest possible in terms of number of dwords bursted, since the maximum cache line size of the Intel 21154 PCI-to-PCI

bridge is 16 dwords [10, sec.15.1.9]. In the FLIC this bridge is between the host system's PCI bus and the FPGA's PCI connection, thus "filtering" cached transactions.

It must also be emphasized that these results were not obtained for the FLIC itself. Surely the FLIC will present its own issues.

Anyway we still present a brief analysis of prospects for further optimizing the read speed.

6.1 Further optimization

- As we know, having one data transferred every clock cycle implies ∼126 MB/s throughput on a 33 MHz 32-bit bus. So taking into account bursts with 16 dwords, and taking into account empirical results, one could achieve 53% of the total bandwidth of the system – ∼67 MB/s – because of the extra clock cycles for the read protocol and the observed 11 IDLE states.
- Since the FLIC is behind one bridge, one expects that, at the beginning of the transaction, this bridge will forward the read request to the FLIC, thus inserting a retry cycle on the host's side. Empirically, one retry transaction and subsequent idle states have been seen to take up to 5 clock cycles. So we can now take at most 45% of the total bandwidth and hope at most for ∼57 MB/s read speed.
- To this we can add the fact that there will be some latency on the FLIC to access it's internal RAM and getting data on the PCI output FIFO. Assuming this latency is of the order of 200 ns, we can estimate that some 3 wait-states on average will be introduced in the bus. This now implies 42% of the total bus bandwidth, or ∼52 MB/s.
- Adding to all these "features" from PCI transactions with real-world devices, one must also bear in mind that the 37.8 MB/s read speed implies *exclusive* usage of the PCI bus, with no other agents using it. Since in real systems this does not happen, namely because of network interfaces needed to take the data out to some concentrator, the read speed will be affected by an yet unknown factor, which should never be smaller than 50%.
  To this last point the discussion on the modern approach of using proprietary interconnect buses between the MCH and the ICH is relevant, since the bandwidth consuming peripherals on the typical DAQ computer will be:

  Hard drive As previously stated, this can take up to 40 MB/s.

  Network controller This should take no more than 1.3 MB/s of bandwidth. But if this component is moved from the PCI bus to a direct connection to the ICH, a lot of transactions on the PCI bus would be saved when sending out data packets.

  The main idea behind this thinking is that the FLICs being *the only* devices on the PCI bus, one might expect a factor of no less than 90% to be applied to the read speed values attained. This would yield an optimized value of 47.3 MB/s.

7 Conclusion

It has been demonstrated, even against AMD's technical staff knowledge/belief, that the CPU can burst read from PCI agents' memories. This highly depends on the chipset used, but the top speed reached in a commodity PC costing less than 2 kCHF was 37.8 MB/s.

The read speed values attained are compatible with the NA60 speed requirements for proton runs, thus alleviating the schedule of the FLIC's software development.

8 Acknowledgements

In order to streamline all the work reported in the few months it took, I have to thank Carlos Lourenço for letting me bite this juicy problem. I also need to thank Hans Muller for repeatedly lending me his newer-than-the-library's PCI books. Also David Dominguez and Angel Guirão helped me, a physicist, to use all the different tools in an electronics lab. They were also responsible for many ideas on what to try next when things weren't getting faster.

A Glossary

With all the jargon used from such diverse areas, such as programming or hardware, including a glossary at the end becomes important. Further doubts on any of these terms or concepts may disappear by simply searching them in Google `http://google.com`.

agent Agent is PCI jargon for a logical device in the bus. A physical device may have itself several functions. Each function will be an agent, for it will have its own PCI configuration space.

Assembly A very low-level programming language, in which the basic constructs are CPU instructions.

ATA100 A standard defining IDE hard drive transfer speeds of up to 100 MB/s.

device A PCI device is a PCI physical device that takes one slot on the bus and represents one electrical load. *E.g.* SBs that have USB and IDE controllers integrated, are one device with two (or more) functions.

dword One double-word. Equivalent to 4 bytes or 32 bits. This is the length of one data unit on a 32-bit PCI bus.

DDR SDRAM Double data rate SDRAM. DDR is achieved by transferring data on both rising and falling edges of the FSB clock.

FPGA Field Programmable Gate Array; A programmable kind of logic, which uses VHDL as its' programming language.

GB Gigabyte. Equivalent to 2^{30} bytes in this note. Some references might use 10^9, thus giving a 7.4% difference.

init The program used to change runlevels in Linux. Besides runlevel 0 which means shutdown, usual runlevels are

1. Single-user mode . No services started, just a shell.
2. Multiuser-mode without NFS.

3. Full multi-user mode.
4. Reserved.
5. X-Windows mode.
6. Reboot.

ISA Industry Standard Architecture bus.

kB kilobyte. Equivalent to 2^{10} bytes in this note.

long `long` type. This is the C type for integers with 32-bit storage.

MB Megabyte. Equivalent to 2^{20} bytes in this note. Some references might use 10^6, thus giving a 4.8% difference.

WBINVD Write-back and invalidate cache. An x86 assembly instruction proven to be very useful in the course of this work. Can only be issued from inside the kernel as `__asm__ __volatile__ ("WBINVD":::"memory");`

word word. Equivalent to 2 bytes or 16 bits.

References

[1] PCI-FLIC — a flexible I/O card for the PCI bus, available at http://hmuller.home.cern.ch/hmuller/FLIC/pciflic.pdf.

[2] ALICE DATE User's Guide, January 2001, CERN ALICE DAQ group, ALICE 00/31 Internal Note/DAQ.

[3] Readout of NA60 detector partitions via PCI-FLIC, available at http://hmuller.home.cern.ch/hmuller/FLIC/partitionro.pdf.

[4] PCI System Architecture, 4th ed., Anderson D. and Shanley T., Addison-Wesley, January 1999 (ISBN 0201309742.)

[5] The Linux 2.2.x Kernel Sources, available at http://www.kernel.org, or under a Linux system on the /usr/src/linux/ directory.

[6] The `pciutils` package information is available at http://atrey.karlin.mff.cuni.cz/~mj/pciutils.html.

[7] Understanding the Linux Kernel, Bovet D. and Cesati M., O'Reilly, October 2000 (ISBN 0596000022.)

[8] AMD-761 System Controller Data Sheet, available at http://www.amd.com/products/cpg/athlon/techdocs/index.html.

[9] Olaf Birkeland, private communication.

[10] Intel 21154 PCI-to-PCI Bridge, Datasheet, available at http://developer.intel.com/design/bridge/datashts/278108.htm.

[11] Intel 815 Chipset Family: 82815EP and 8281P Memory Controller Hub (MCH), Datasheet, available at http://developer.intel.com/design/chipsets/datashts/290693.htm.

[12] The IA-32 Intel Architecture Software Developer's Manual, Volume 2: Instruction Set Reference, available at http://developer.intel.com/design/pentium4/manuals/245471.htm.

[13] The IA-32 Intel Architecture Software Developer's Manual, Volume 3: System Programming Guide, available at http://developer.intel.com/design/pentium4/manuals/245472.htm.

A Plug and Play Approach to Data Acquisition

J. Toledo, H. Müller, J. Buytaert, F. Bal, A. David, A. Guirao, and F. J. Mora

***Abstract*—Backplane buses are becoming a legacy for high-rate, high-volume data processing applications. Higher efficiency at lower cost is offered by the PCI bus technology, compared to crate-embedded processors. Becoming part of the plug&play domain of the host's operating system, no additional data transfer protocols are needed. We have combined the PCI technology with high-density field-programmable gate array (FPGA) logic and common mezzanine standards on a flexible PCI card. First applications cover readout controllers for legacy bus protocols, high-speed link I/O and fast analog input data conversion. An FPGA with embedded PCI master/target core serves as programmable interface between the PCI bus, mezzanine cards, and a local SDRAM. Adapter mezzanine cards, implemented according to the IEEE P1386 or similar common standards, are used for voltage level conversion, trigger interfacing or preprocessing. The application-dependent controller functions as well as SDRAM and PCI interfacing are handled by FPGA logic. A Linux driver was developed to achieve high bandwidth via CPU-initiated transfers. Control software for Windows and an interface for LabView target control and monitoring applications via graphical interfaces. First experience and applications will be reported.**

***Index Terms*—Data acquisition buses, detector readout architectures, DMA transactions, PCI bus.**

I. Introduction

THE rack-mounting backplane bus standards like VME ruled the HEP (High Energy Physics) data acquisition (DAQ) arena during the last two decades, with practical transfer rates across the backplane in the 10-to-50 MB/s range and inter-crate communications limited to 3 MB/s (VICbus for VME) and 40 MB/s (Fastbus segment interconnects). Switched networks for event building in the GB/s range replaced inter-crate buses in the late nineties. But the conventional method of "data processing across the backplane" is still in use.

DAQ modules in today's experiments must each handle throughputs of up to 200 MB/s, resulting in an integrated bandwidth across the backplane far higher than what VME64 and other recent VME developments can provide.

PCI invaded the VME bus since 1995 via standards like PMC (PCI Mezzanine Card, IEEE P1386.1-1995), which is a PCI implementation of the CMC (Common Mezzanine Card, IEEE P1386) form factor, allowing both 32- and 64-bit PCI implementations. PMC cards added a large degree of freedom, allowing drop-in integration of any device or technology which could be adapted to the PMC standard, like embedded CPU's, interfaces for network standards and high-sped links.

PMC also provided the possibility for non-PCI communication between the mezzanine and the board logic via a user-defined 64-pin input/output (I/O) connector. Typical examples for use of this user-defined I/O connector are serial buses like USB, I2C or JTAG. Motherboard-resident control chips which require remote configuration functionality can herewith be easily configured from mezzanine-resident, networked processor systems.

Extensions to the PMC mechanical standard like the VITA-32 standard for embedded processors added the possibility to place a PCI host system on a mezzanine. This removed the necessity for a PCI host system on the motherboard.

Manuscript received November 1, 2001; revised February 14, 2002.

J. Toledo and F. J. Mora are with the Department of Electronics Engineering, Universidad Politécnica de Valencia, Spain, Escuela Politécnica Superior de Gandia, Ctra. Nazaret a Oliva s/n, Grao de Gandia, 46730 (Valencia), Spain. (e-mail: jtoledo@eln.upv.es; fjmora@eln.upv.es).

H. Müller, J. Buytaert, F. Bal, A. David, and A. Guirao are with CERN, Geneve 23, CH-1211, Switzerland. (e-mail: Hans.Muller@cern.ch; Jan.Buytaert@cern.ch; Francois.Bal@cern.ch; Andre.David@cern.ch; Angel.Guirao@cern.ch).

Publisher Item Identifier S 0018-9499(02)06135-X.

II. The PCI Bus in HEP Applications

Confined to short length and delicate in its electrical properties, PCI posed sometimes problems. Also the uni-directional PCI bus hierarchy, requiring a host for initialization at the root segment, was a constraint that limited flexibility. Nevertheless, PCI offered a fast, processor independent, industry supported solution and its widespread use in HEP DAQ systems became a fact predicted in the mid nineties [1], [2]. Practical implementations followed two paradigms described in the next sections.

A. On-Board Data Processing

Data are actually more efficiently processed locally (migrating from "data processing across the backplane" to "data processing across the on-board bus") and I/O is de facto using today's high-speed link technologies. This relegates VME and Fastbus to the role of convenient mechanical frameworks with a "power and control bus connector" rather than providing bus functionalities via backplanes.

This new paradigm relies on the use of PCI as local DAQ bus and on the use of mezzanine card standards for exchangeable I/O interfaces. In the conservative approach, the backplane was left in the crate in order to maintain backward compatibility, but the PCI bus was integrated into the DAQ module logic for high-speed chip-to-chip communication. This possibility was enhanced by the availability of good PCI support chips and FPGAs with PCI port.

FPGA technology backed this trend by boosting local processing with FPGA-based coprocessors and easing the integration of data processing and interface logic in small mezzanine form factors. The implementation of specialized algorithms in an FPGA using only integer and boolean operations, in comparison to a CPU, can result in a large performance gain [3]. Data compression, track finding, trigger algorithms, and subevent building can be implemented either on FPGA-based coprocessor mezzanines, assisting a CPU on the main board, or entirely in FPGAs.

The advent of large-gate-count FPGAs allowed for the integration of data processing algorithms, small to medium-sized FIFOs, SRAMs, and extra glue logic in a single device. The availability of FPGA-embedded cores (hard and soft) like PCI master/target interfaces, HSL (High-Speed Link) serializers, DSPs, etc., provides very powerful data processing and I/O functionalities in a single device.

As a result, complex systems can be embedded in a small mezzanine card. Recent examples can be found in PMCs for real-time data processing [4], in the S-Link specification for link technologies on CMC mezzanine cards [5] and ADC cards with FPGA control and processing [6].

A recent implementation of this paradigm is the Readout Unit for the LHCb experiment [7], a 9U-sized card with four CMC mezzanines in the front panel for data input, on-board PCI bus for FPGA-based event building and a PMC network interface card on the rear for data output. Hybrid solutions like the one described in [8] use front-panel links for data input, mezzanines for data processing, PCI for on-board mezzanine interconnection and VME64 for board readout.

B. Data Processing Across the PCI Backplane

The second paradigm consists in replacing VME by PCI as DAQ backplane. Some PCI derivatives, like the CompactPCI standard appeared, but the acceptance was mild. The PC was a more successful (and cheaper) PCI readout platform, providing also the ground for what we call the flexible I/O concept.

III. The Flexible I/O Concept

When contemplating the trends around the VME bus legacy, one realizes that a rearrangement of the building blocks (i.e., mezzanine cards, FPGA's, PCI bus, user connector, memory) can result in a new, flexible concept. For example, by moving PMC mezzanine cards directly onto standard PCI cards inside a host PC, by mere addition of a simple bridge function, the PMC becomes part of the plug & play devices seen by the operating system of the readout computer. The PC's power supply, PCI backplane, and mechanical housing replace the crate at a lower cost. The host PCI bus is naturally also available for slow control tasks. With such a configuration, no more intermediate data transfer mechanisms between an external bus controller and the host are needed.

The use of PMC inside PCs is not foreseen by the P1386 standard. However, the PCI connector pitch on an ATX motherboard corresponds to the pitch of modules in a VME crate and modern PCs can power up to six PCI cards. Hence there is neither a mechanical, nor logical constraint to not use PMCs in a PC.

We went one step further. By the addition of FPGA and SDRAM on the PCI card, the possibility for implementing flexible readout control functions is available, however localized in the readout computer. By adding memory in some multipurpose configuration, all ingredients are available for designing typical data acquisition readout applications on a standard PCI plug & play card. The result is the flexible I/O concept, where mezzanines are used as adapters for voltage level conversion, preprocessing, and trigger interface, and FPGA logic is used for readout control and as interface to PCI and RAM. The flexible I/O concept is depicted in Fig. 1. References [9]–[12] describe PCI cards approaching the flexible I/O concept.

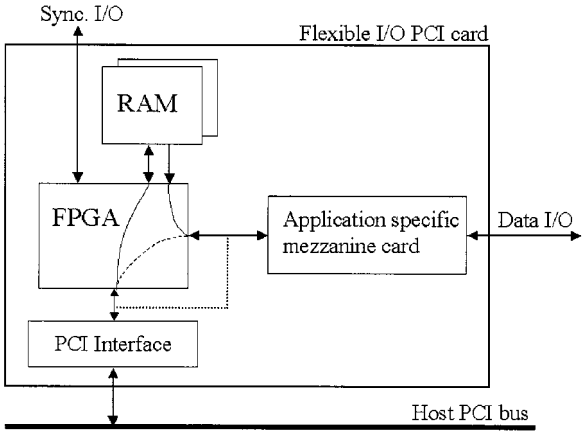


Fig. 1. Main components of a flexible I/O PCI card. The PCI interface can be integrated in the FPGA saving board space. Reconfigurability allows a large number of features like swing buffers in RAM, data processing before and/or after storage in RAM, external synchronization signals or direct access to PMC mezzanines from the host PCI bus.

Immediately apparent applications cover I/O via high-speed links with FPGA-based coprocessor [10], [13], readout controllers for legacy bus protocols [14] and chip-based detector readout like Si-pixel [15] or Si-strip [16].

A. Moving Data Across the PCI Bus

In the flexible I/O concept, the PCI expansion bus in the host PC is effectively the DAQ backplane. All devices on a PCI system share the same memory space, so a transaction to the host main memory, to the video card memory or to a DAQ board's buffer only differs in the destination address. DMA is the natural data transfer mode in PCI, which uses burst operations.

Depending on the configuration (32/64 b, 33/66 MHz) the raw bandwidth in a PCI bus ranges from 132 to 528 MB/s for infinite-length bursts [17]. The shorter the burst, the lower the performance due to PCI protocol overhead. As an example, a single-word transaction takes a minimum of four cycles, yielding a maximum of 33 MB/s on a 32-b 33-MHz bus. Thus, the key to high performance is to perform long bursts. This implies using always PCI memory space, as no burst transactions are allowed for PCI I/O space. Two different scenarios have been tested.

1) Bus-Mastering DMA: In this case the DAQ board implements a PCI interface capable of acting as initiator and performing DMA write transactions to host memory. The burst length is only limited by the target (i.e., the PCI chipset's interface to main memory) and by PCI arbitration algorithm in the PCI chipset. Thus, the performance is platform dependent.

In the past, PCI (32 b, 33 MHz) exceeded the host's RAM bandwidth (see PC-A plot in Fig. 2), but the advent of PC-100 and PC-133 SDRAM (see PC-B plot in Fig. 2) allowed higher throughput and justified the use of faster PCI implementations (64-b and 66-MHz).

2) PCI Read/Write With the CPU: Performance tests moving data blocks larger than 64 KB to and from an ATI 3D RAGE PRO video card have been carried out [19] on several

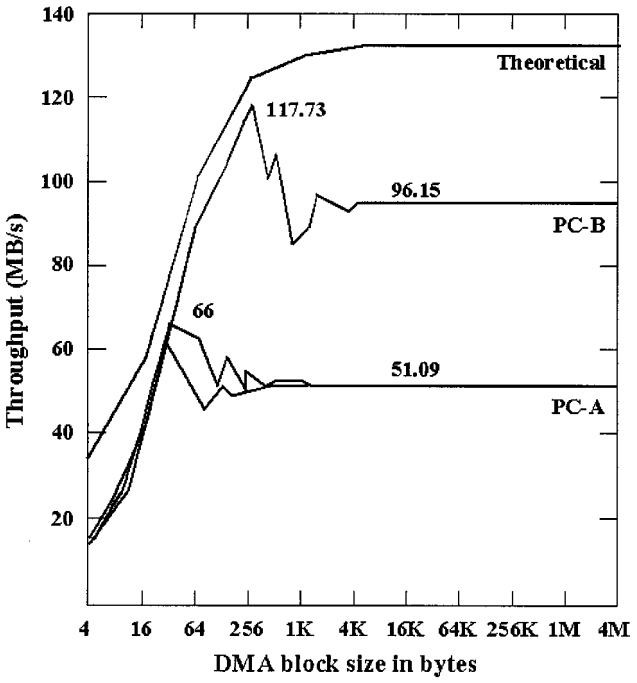


Fig. 2. Throughput measurements across a 32-b 33-MHz PCI bus for two different platforms as a function of the DMA block size. A PCI card based on the AMCC S5933 PCI interface performs DMA write operations to host main memory [18]. PC-A is based on the now obsolete i430HX chipset, 100-MHz Pentium CPU, 60-ns EDO RAM and video card on the PCI bus. PC-B is based on the i440BX chipset, 400-MHz Pentium II CPU, PC-100 SDRAM and AGP video. No graphical user interface is used in order to provide a fair comparison. Peak performance for PC-A is limited to 66 MB/s by the 32-b 60-ns RAM; whilst in PC-B the limiting factor is the maximum PCI burst length (256 byte). Performance decrease in PC-A is due to PCI retry operations inserted by the chipset when its internal buffers get full. Performance drop in PC-B is explained by the fact that DMA blocks are split into a number of maximum-size bursts spaced by retry operations.

platforms running Linux 2.2.x. Test utilities were executed in kernel mode for higher performance.

Write tests reported 103 MB/s on a 400-MHz Pentium II with i440BX chipset, 112 MB/s on an 800-MHz Pentium III with i815EP chipset, but only 63 MB/s on an Athlon-based PC with VIA-KT133 chipset. All platforms had 32-b 33-MHz PCI.

Read tests, on the other hand, did not yield more than 9 MB/s in any of the tested platforms. This is explained by the fact that PCI chipsets are not optimized for CPU reads from PCI, as bus-mastering DMA is a more efficient paradigm to handle this data flow.

Nevertheless, read performance can be boosted as described in detail in [19]. First, the PCI board's prefetchable memory must be defined as cacheable in the chipset's Memory Type Range Registers (MTTR) [20] in order to force a cache fill request rather than a data request. This yielded a 21-MB/s performance on the i815EP platform, no improvement at all on the VIA-KT133 and 26.2 MB/s on an AMD761-based platform. Second, the cache line size register in the PCI bridge in the PCI board (an i21154 PCI-to-PCI bridge in our tests) was set to the maximum value (16 32-b words), allowing longer bursts and yielding 37.8 MB/s on an AMD761 platform.

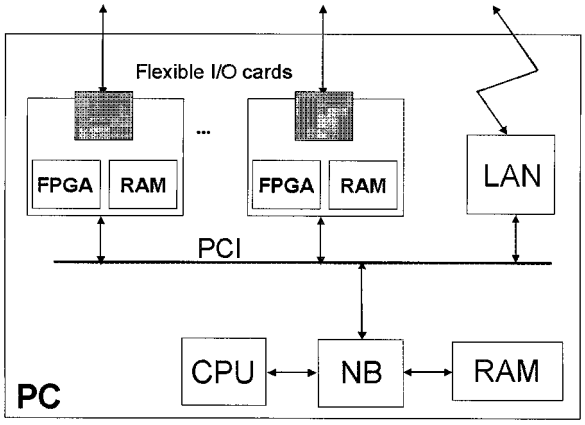


Fig. 3. Flexible readout architecture based on a commodity PC, flexible I/O cards with application-specific mezzanines and a LAN interface for slow control. Scalability is limited by the number of PCI slots on the motherboard (typically five), the PCI bus bandwidth and the processing requirements in the CPU.

B. Readout Architecture

The bandwidth limitations discussed in the previous sections must be considered in a wider context. In the typical scenario (Fig. 3), several PCI DAQ boards and a LAN card (interface to the control system) share the PCI bus bandwidth, while the DAQ software is running on the host CPU further degrading system performance. Such a scenario, but for a single DAQ card, has been evaluated in [21].

The evolution in motherboard technology, like independent PCI segments for traffic isolation and 64-b 66-MHz PCI buses, can enhance scalability in high-throughput applications.

IV. The PCI-FLIC Card and Its Applications

Our flexible I/O card implementation, the PCI-FLIC (see Fig. 4), is described elsewhere [11]. Compared to other flexible I/O cards, all data paths are 64 b wide, including the PC-100 SDRAM and the PMC connector. Combined with 66-MHz PCI bus capability, the PCI-FLIC is well suited for very high throughput applications. Different readout architectures (dual-port memory, FIFOs, circular buffers and swing-buffers of different width up to 64 b) are supported. Data processing may be implemented in FPGA or on mezzanine.

A. Applications

1) NA60 Data Acquisition System: The readout of NA60 muon spectrometer's multiwire chambers (MWC's) relied on an old CERN system [23] consisting of: 1) 32-channel readout CAMAC modules grouped in up to 22 modules per crate; 2) a RMH system encoder collecting hit data via a CAMAC branch cable and outputting to a custom ECL cable according to a protocol named RMH; 3) a VME memory module into which events are written by the RMH encoders; and 4) a VME interface to a readout computer. The 1.6-kHz trigger rate and event sizes in the order of 1 KB require 1.6-MB/s throughput. To increase the event efficiency during 10 s of machine spill, the bandwidth had to be increased to 6 MB/s, which required faster RMH handshake and a buffer that was four times larger.

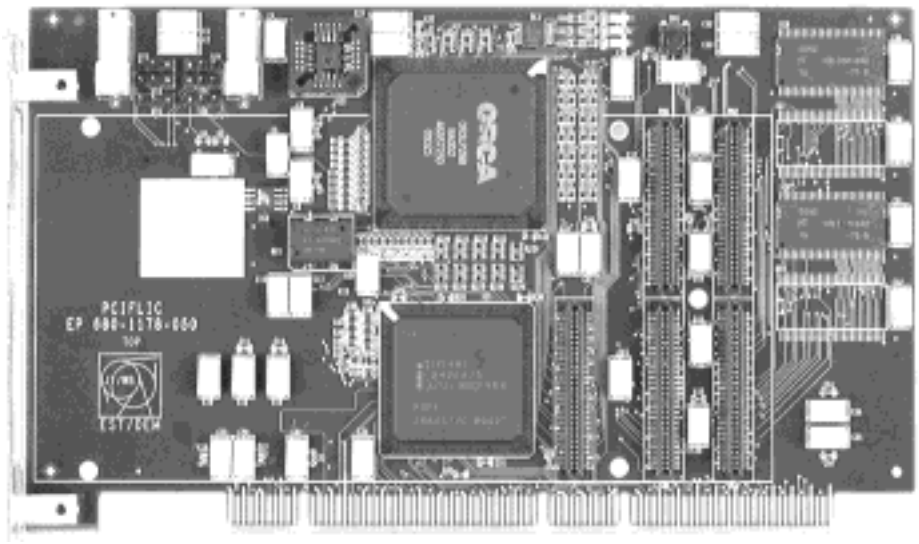


Fig. 4. The PCI-FLIC card is a general-purpose 32/64-b 33/66-MHz universal PCI card with connectors for a variety of mezzanine adapters, centered on a 120-kilogate FPGA (center, top) with an embedded PCI master/target core. This PCI card contains two banks of 32-MB SDRAM (right) and two general-purpose LVDS I/O plugs (top left corner). The FPGA logic serves as programmable interface between the connectors, the PCI bridge (center, bottom) and local SDRAM. There are five mezzanine connectors for PMC-32 (including user defined I/O connector), 64-b PMC extensions and custom mezzanines like S-Link. Estimated power dissipation is 15 W including a mezzanine card (limited to 7.5 W according to [22]). PCI 2.2 allows up to 25 W per PCI card.

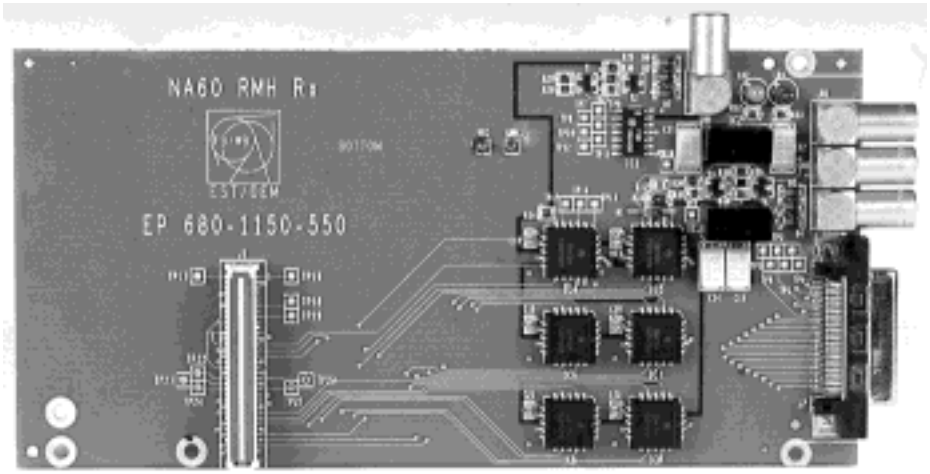


Fig. 5. RMH mezzanine for the PCI-FLIC. NIM connectors are used for the RMH trigger interface (top right corner). The RMH cable is connected via a 50-pin connector (bottom right). Signals are converted from ECL to TTL levels before being transmitted to the PCI-FLIC via a standard 64-pin CMC connector (left).

A mezzanine card has been designed [14] to adapt the RMH cable and ECL electrical levels to the PCI-FLIC and thus replace to the VME memory and interface modules with a faster RMH receiver with larger buffer. The buffer is interfaced via PCI to the readout computer (the host PC).

The design of different mezzanine cards to read out the other NA60 sub-detectors is planned, requiring a total of eleven PCI-FLIC cards with their corresponding mezzanines (one per sub-detector partition) housed in four host PCs (local data concentrators). In the proposed scheme, 150 MB of partition data per machine spill are sent to a Global Data Concentrator PC via a Fast/Gbit Ethernet switch, assembled into complete events and routed to a remote disk server via the Ethernet switch.

The readout of the muon spectrometer has been successfully implemented using PCI-FLIC and RMH mezzanine cards and the L188 driver described in Section IV-B, yielding a 37.8-MB/s throughput. Higher performance can be obtained with a bus-mastering DMA scheme.

2) Other Applications: In another application, a PC-based test station, we use the PCI-FLIC card with an 80 MB/s S-Link receiver/transmitter mezzanine for transmitting and receiving test data over a twisted pair.

B. Software for the PCI-FLIC Card

Using a commodity PC as host, two major environments can be targeted: Linux and Windows.

1) Linux Environment: A Linux 2.2.x driver allowing up to five PCI-FLIC cards was developed by the NA60 experiment. It supports CPU-initiated data transfers to and from the PCI-FLIC SDRAM using the optimizations mentioned in Section III-A-II. A utility was developed to directly reconfigure the FPGA from the host CPU via the PCI bus, allowing (in a networked PC) remote reconfiguration.

2) Windows Environment: A Dynamic Link Library (DLL) for Windows 98/2000/NT was developed using Jungo's Windriver utility [24]. This library provides functions to access the memory and configuration registers in the FCI-FLIC card and can be used by higher-level software layers. An interface for LabView has been developed based on this DLL, allowing control and monitoring applications via graphical interfaces.

V. Conclusion

The PCI bus available in off-the-shelf PCs offers a high-performance, cost-effective alternative for conventional crate-based DAQ systems. Among different possibilities, the use of commodity PCs housing general-purpose FPGA-based PCI cards with application-specific CMC mezzanines is considered. This results in a flexible I/O concept with has a number of benefits like superseding VME bandwidth limitations.

We developed a flexible I/O card (PCI-FLIC) and the RMH mezzanine for the CAMAC-based NA60 muon spectrometer readout. Both cards are being used successfully in the experiment (see Fig. 5).

Similar approaches to the PCI-FLIC have been developed independently for similar applications, however sometimes lacking general-purpose I/O, limiting I/O to 32 b, or having less or no support for buffered architectures.

References

[1] A. van Praag *et al.*, "Overview of the Use of the PCI Bus in Present and Future High Energy Physics Data Acquisition Systems,", Santa Clara, CA, Tech. Rep. CERN/ECP 95-4, 1995.

[2] H. Müller, A. Bogaerts, and V. Lindenstruth, "A millenium approach to data acquisition: SCI and PCI," in *Proc. Int. Conf. on Computing in High-Energy Physics: CHEP'95*, 1995, pp. 388–393.

[3] C. Hinkelbein *et al.*, "Pattern recognition algorithms on FPGA's and CPU's for the ATLAS LVL2 trigger," *IEEE Trans. Nucl. Sci.*, vol. 47, pp. 362–366, Apr. 2000.

[4] J. V. Pascual, V. González, E. Sanchis, G. Torralba, and J. Martos, "Development of a DSP-based PMC board for real time data processing in HEP experiments," in *Proc. 12th IEEE NPSS Real Time Conf.*, 2001, pp. 235–239.

[5] H. C. van der Bij, R. A. McLaren, O. Boyle, and G. Rubin, "S-Link, a data link interface specification for the LHC era," *IEEE Trans. Nucl. Sci.*, vol. 44, pp. 398–401, June 1997.

[6] S. A. Baird, J. A. Coughland, R. Halsall, J. Hartley, W. J. Haynes, and T. Parthipan, "A PMC based ADC card for CMS tracker readout," *IEEE Trans. Nucl. Sci.*, vol. 47, pp. 158–161, Apr. 2000.

[7] J. Toledo, F. Bal, D. Domínguez, A. Guirao, and H. Müller, "A readout unit for high rate applications," *IEEE Trans. Nucl. Sci.*, vol. 49, pp. 448–454, April 2002.

[8] M. J. Le Vine, A. Ljubicic Jr., M. W. Schulz, R. Scheetz, C. Cosiglio, D. Padrazo, and Y. Zhao, "The STAR DAQ receiver board," *IEEE Trans. Nucl. Sci.*, vol. 47, pp. 127–131, Apr. 2000.

[9] O. Brosch *et al.*, "MicroEnable, a reconfigurable FPGA coprocessor," in *Proc. 4th Workshop Electronics for LHC Experiments*, 1998, CERN/LHCC/98-36, pp. 402–406.

[10] M. Drochner, W. Erven, M. Ramm, P. Wüstner, and K. Zwoll, "A VME controller for data acquisition with flexible gigabit data link to PCI," in *Proc. 12th IEEE NPSS Real Time Conf.*, 2001, pp. 204–207.

[11] H. Müller, F. Bal, A. David, D. Domínguez, and J. Toledo, "A flexible card concept for data acquisition: The PCI-FLIC," in *Proc. 12th IEEE NPSS Real Time Conf.*, 2001, pp. 259–253.

[12] O. Brosch. (2001) RACE-1 (reconfigurable accelerator computing engine). Univ. Mannheim. [Online]. Available: http://www-li5.ti.uni-mannheim.de/fpga/

[13] R. Bock, J. A. Bogaerts, P. Werner, A. Kugel, R. Männer, and M. Müller, "The active ROB complex: An SMP-PC and FPGA based solution for the ATLAS readout system," in *Proc. 12th IEEE NPSS Real Time Conf.*, 2001, pp. 199–203.

[14] H. Müller, A. David, D. Dominguez, M. Floris, S. Popescu, and P. Rosinsky. (2001) Readout of NA-60 Detector Partitions via PCI-FLIC. [Online]. Available: http://hmuller.home.cern.ch/hmuller/FLIC/partitionro.pdf

[15] G. Usai, "The Pixel Readout Board," CERN, NA60 Tech. Note.

[16] A. David, J. Buytaert, J. Lozano, and R. Shahoyan, "Readout of Na60's Silicon Strip Planes," CERN, NA60 Tech. Note.

[17] "PCI Local Bus Specification. Revision 2.2," PCI Special Interests Group, 1998.

[18] P. Costi, "Evaluación de la tecnología de bus PCI para el desarrollo de tarjetas de adquisición de datos de altas prestaciones," Master Thesis, Universidad Politécnica de Valencia, Spain, 1999.

[19] A. David. (2001) Performance Issues in PCI Reading. [Online]. Available: http://adavid.home.cern.ch/adavid/public/NA60/online/PCIReadingNote/PCIReadingNote.pdf

[20] *The IA-32 Intel Architecture Software Developer's Manual*, vol. 3, System Programming Guide, Intel.

[21] W. Carena *et al.*, "PCI Based Read-Out Receiver Card in the ALICE DAQ System. Version 1.0.," CERN, ALICE-PUB-2001-47, 2001.

[22] *Draft Standard for a Common Mezzanine Card Family: CMC*, IEEE Std. 1386 Draft 2.0, Apr. 1995.

[23] J. B. Lindsay, C. Millerin, J. C. Tarle, H. Verweij, and H. Wendler, "RHM: A fast and flexible data acquisition system for multiwire proportional chambers and other detectors," in *Wire Chamber Conf.*, Vienna, Feb. 1978.

[24] Jungo. Jungo's Home Page. [Online]. Available: http://www.jungo.com

A New PCI Card for Readout in High Energy Physics Experiments

Michele Floris, Corrado Cicalò, Davide Marras, Gianluca Usai, and André David

***Abstract*—Recently, some high energy physics experiments have started to adopt readout systems based on the PCI architecture. In this context, a new PCI card that can be adapted to several readout schemes has been designed. It contains a 64-MiB[1] local buffer, programmable FPGA logic, a hardware PCI bridge, and can be connected to mezzanine cards. The card is presently used in the NA60 experiment at the CERN SPS for the readout of several detectors. The interfacing to the different front-ends is provided by the mezzanine cards while the readout protocols are implemented in the FPGA. Moreover, it is used as a test readout system for the ALICE experiment muon chambers. This paper describes the card, with particular emphasis on its flexibility and relatively simple development, related to the use of an external PCI bridge.**

I. Introduction

THE PCI bus offers very attractive features for readout applications in high energy physics: It allows for good performance (bandwidth up to some 100 MiB/s in master mode) at a relatively low price. Moreover, interfacing with the DAQ software is extremely easy, as discussed below.

The development of a PCI device can be greatly simplified by using hardware bridges. These devices take care of all the complexities of the PCI bus and the user can implement simplified transactions on a simplified bus.

By combining these cores with FPGA technology, it is possible to obtain a very flexible solution, which can be adapted to several different schemes.

In this paper, a PCI card which has been developed following these ideas is presented. This is presently used in several different readout applications.

In Sections II–IV, the target application and requirements for the card development, the actual implementation, and performance are described. Finally, all the applications where this card is presently used are listed.

II. Motivation and Requirements

The drive toward the development of the card was the idea to use it as the basic unit in the readout of the NA60 experiment at the CERN SPS [1]. NA60 is a heavy ion experiment which is currently in the data-taking phase. Its apparatus consists of four detectors (a muon spectrometer, a zero degree calorimeter, a silicon strip beam tracker, and a silicon pixel vertex telescope) with very different front-end electronics. The idea was to read all of them using the same general purpose PCI card. The adopted strategy is to use detector-specific mezzanine cards to interface the different front-ends. It was also decided to divide the apparatus into several independent partitions, from the readout point of view.

At the CERN SPS particles do not hit the target continuously but rather during time intervals called "bursts" or "spills," typically 5 s long, separated by "interbursts," typically 10 s long. This is an important constraint to take into account in developing a DAQ system and the readout of the NA60 experiment is indeed spill buffered. This means that the data from each partition are read out independently and stored in a local memory during the burst, when they are produced. Data from different partitions are read by the software during the interburst to be stored on tape [2].

In summary, the card requirements were:

1) *flexibility*, as it was meant to be used for all the detectors;
2) *compatibility with mezzanine cards*, to interface the different front ends;
3) *on board memory*, to be used as a local buffer;
4) *fast development*, as NA60 is a running experiment;
5) *low cost*.

III. PCI-CFD Card

Fig. 1 shows the card which has been developed, called PCI compact and flexible design (CFD). It is built around an FPGA, which is the natural choice in order to have a flexible solution. The FPGA [Fig. 1(a)] is an Altera APEX EP20K100, which has enough logic gates for our purposes (about 250 000), in a BGA package, having a reasonable number of I/O pins (about 240) in a limited space.

The PCI interfacing is handled by a PLX hardware bridge [Fig. 1(b)]. This device takes care of all the complexities of the PCI bus and provides a local TTL bus, where the user can implement transactions by using a simplified protocol. This chip can be customized (for example PCI memory spaces can be defined) and the configuration parameters are stored in an EEPROM which can be written from PCI (see Section III-B). The bridge used is a PLX 9030 [3], which works as a PCI target. More than satisfactory performance (see Section III-C) can be achieved

Manuscript received November 29, 2003; revised April 29, 2004 and July 2, 2004. The work of A. David was supported in part by the Portuguese Fundaçao para a Ciência e Tecnologia under Grant SFRH/BD/4761/2001 and in part by the CERN Doctoral Student Programme.

M. Floris and G. Usai are with Università degli Studi di Cagliari 09042, Italy, and also with the INFN, Cagliari 09042, Italy.

C. Cicalò and D. Marras are with the INFN, Cagliari 09042, Italy.

A. David is with CFIF-IST, Lisbon, Portugal, and also with CERN, Geneva, Switzerland.

Digital Object Identifier 10.1109/TNS.2004.835567

[1]In this paper, we comply with the standard IEC 60027-2 Ed. 2.0 (2000–11) "Letter symbols to be used in electrical technology—Part 2: Telecommunications and electronics."

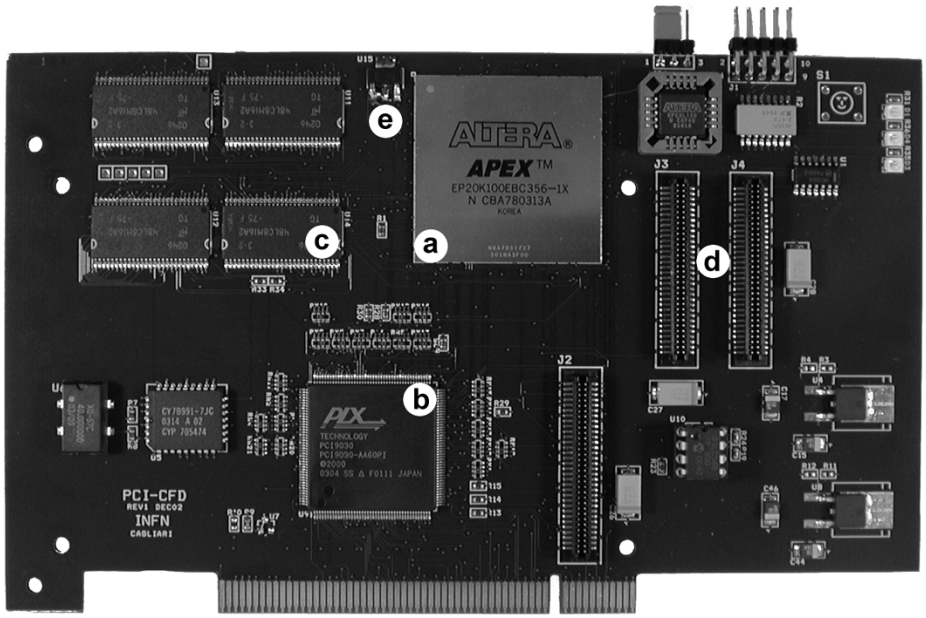


Fig. 1. PCI CFD card. (a) FPGA. (b) PCI bridge. (c) SDRAM. (d) Mezzanine card connectors. (e) Silicon serial number.

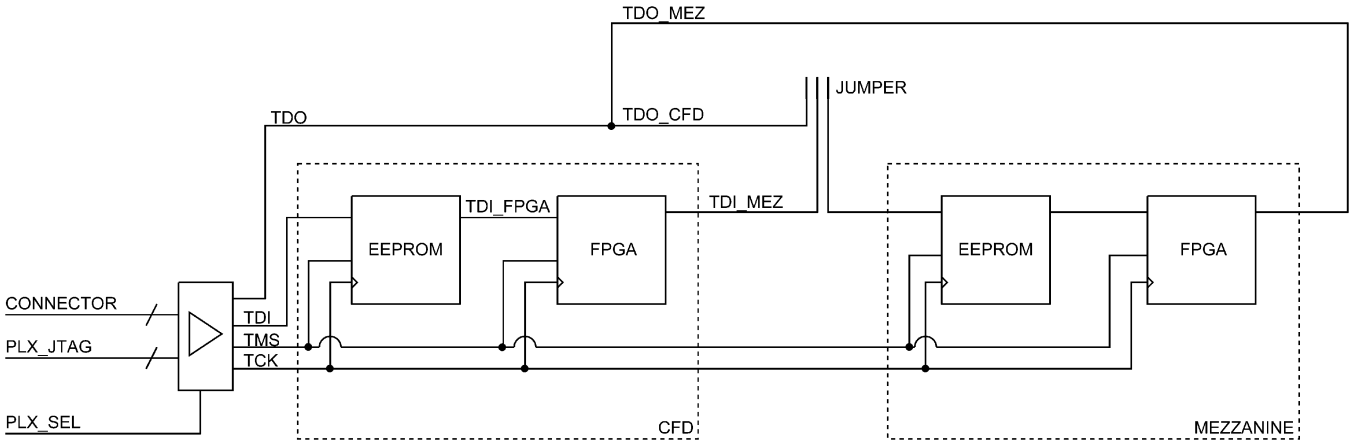


Fig. 2. JTAG chain as implemented in the PCI CFD card. The Altera FPGA can be programmed directly from PCI and the JTAG chain can be extended to the mezzanine.

with this device, with less complication with respect to a PCI master.

The board is equipped with 64 MiB of SDRAM [four banks, 16 MiB each, Fig. 1(c)], which are used as the spill buffer, and with connectors for mezzanine cards [Fig. 1(d)]. The mezzanines are based on the IEEE standard for CMC cards [4] (with two connectors) and the card is compliant with the recommended physical arrangement for S-Link cards [5]. On the other hand, it is not compliant with the IEEE standard for PMC cards [6]. While this standard foresees the possibility to extend the PCI bus to the mezzanine, this feature is *not* implemented in our card, because it was an unnecessary complication for our purposes.

A silicon serial number [DALLAS DS2401, Fig. 1(e)], allowing to identify every PCI CFD uniquely, is also present. This is a very useful feature when more than one card is plugged on the same bus.

Another interesting feature of the card concerns the JTAG [7] configuration of the Altera FPGA (Fig. 2). Usually, these FPGAs (and the EEPROMs which store their configuration) are configured through a JTAG interface implemented by plugging a special cable (which is typically connected to the PC parallel port) to a dedicated connector on the board and using a special software from Altera. As for our application, some general purpose I/O pins available on the PLX chip were used to implement a PCI JTAG interface. In this way, it is possible to configure the FPGA via the PCI bus, without having to physically access the device or plug any extra cable. An existing software was modified for this purpose (see Section III-B). The source of the JTAG stream (PLX/PCI or connector) can be selected via a tristate buffer, which is also controlled by one of the PLX general purpose I/O pins. Moreover, it is possible to extend the JTAG chain to the mezzanine card using a jumper. In this way, FPGA devices (both on the PCI and on the mezzanine card) can be reprogrammed on the fly in a straightforward way. This feature is very useful for DAQ applications, where some of the cards may be located in the experimental zone and thus not easily accessible.

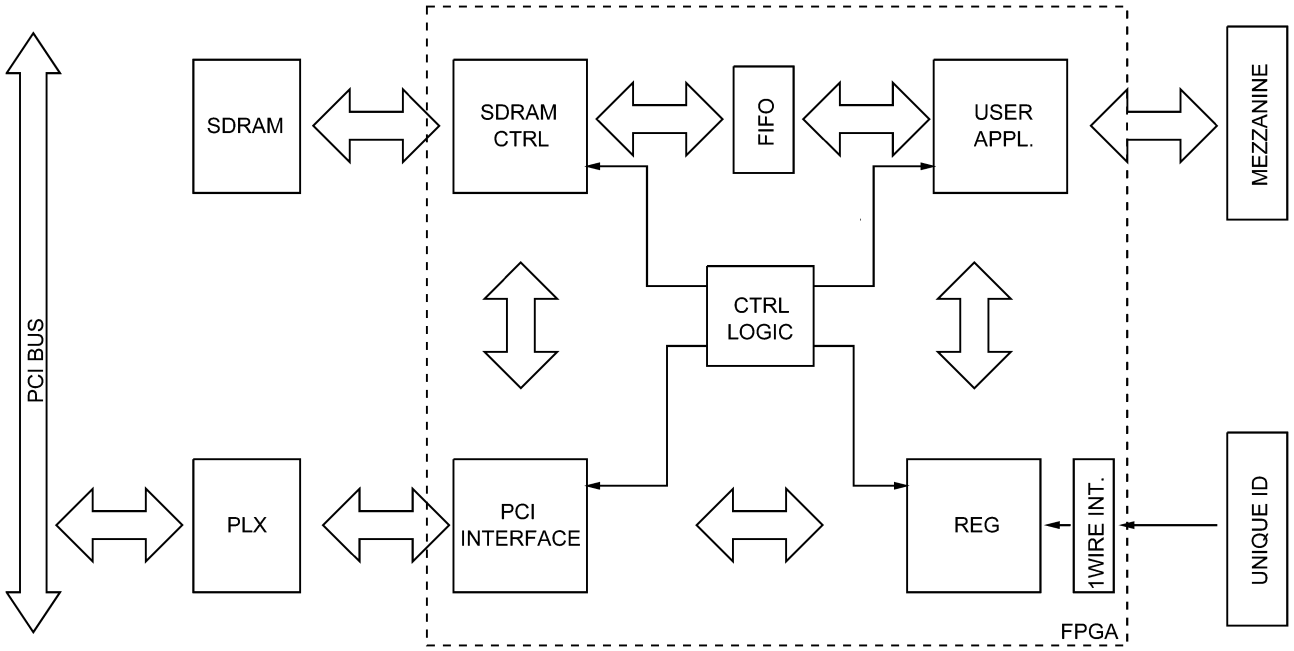


Fig. 3. Block diagram of the PCI CFD card and of the logic implemented inside its FPGA.

The development time for this card was very short (about two months), mainly thanks to the PLX chip which made the PCI interfacing extremely simple.

A. Card Implementation

Fig. 3 shows a block diagram of the CFD card, expanding the logic which is implemented inside the FPGA. It consists of five main blocks: SDRAM controller, PCI interface, register block, user application and control logic.

The SDRAM controller (SDRAM CTRL, Fig. 3) provides a simplified interface to the SDRAM and sends all the needed control sequences (the refresh commands, for example). A Verilog/VHDL SDRAM controller, freely available from Altera, was embedded in the design, shortening the development time of the FPGA logic. In our application, the SDRAM controller is configured to perform reads/writes in an eight-word burst mode. It has been complemented with a FIFO which caches the necessary number of words before starting the transactions. The data words are 32 bit wide, to cope with the PCI bus width. As the SDRAM physical banks have a 16-bit data bus, the highest 16 bits are written in a bank and the lowest in another. Two memory banks are then cascaded to get the desired memory size, the address bus, therefore, being 24 bits wide.

The PCI interface is used to implement the transactions on the local bus with the PLX. This chip is set to work with eight-word-long bursts as well, in order to match the SDRAM.

The registers block (REG) contains some general purpose registers which can be accessed via PCI. They are used to store information about the readout (e.g., number of events, number of bursts, etc.) and to implement the handshake between software and hardware [2]. The contents of the unique identifier (the silicon serial number) are read through a one-wire interface and written in a PCI register so that they can be accessed by the software.

In some applications, it is necessary to send commands to the on-detector electronics, in which case, complex mezzanines have to be used. A serial link to send commands from the CFD to the mezzanine has been implemented. The command to be sent is written to a PCI register by the software and then a second register is accessed to trigger the transmission of the command through the serial link. Some commands might produce an output, which is written to an "OUTPUT" register and can, thus, be accessed by the software.

The user application is the interface with the mezzanine. It is used to implement the detector-specific protocol to get the data from the front-end, format them, and store them in the SDRAM.

Both the user application and the PCI interface can access the memory and registers. For this reason, a control logic (CTRL LOGIC) is needed to arbitrate accesses to the memories. In the case of NA60, this is the same for all detectors and it is fairly simple: access is granted to the user application during the burst, when events are produced, and to the PCI interface during the interburst, when events need to be read out by the software. In a different environment, a different policy to access the memories, and, thus, a different control logic might be needed.

B. Software

Some support software for the Linux operating system has been developed. It includes a program to write the EEPROM of the PLX 9030, which is used to change the configuration of the PLX without an external EEPROM programmer. The vendor only provided a similar program for the Windows operating system.

An existing software (jamplayer [8]), used to JTAG-program the Altera devices using a special cable, has been modified in order to allow the PCI configuration of these devices. This program uses a standard file format (jam [8]).

A driver for the 2.2 Linux kernel has been developed. The PCI memory is seen by the operating system as a kind of 193 ension"

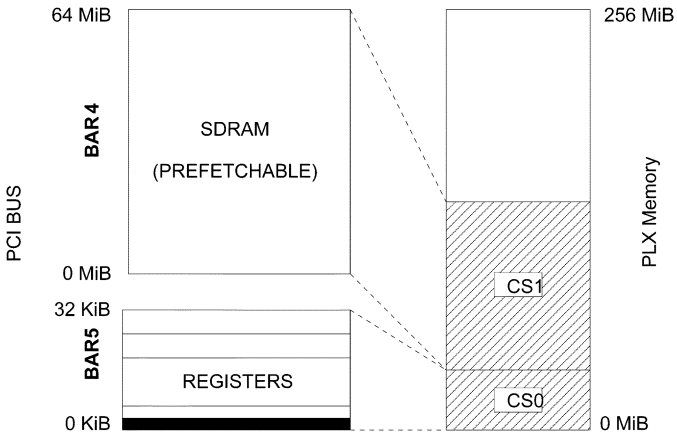


Fig. 4. Map of the PCI-CFD memory. 32 KiB are reserved for the registers, but only 64 B (16 registers, in black) are used by existing applications. 64 MiB are reserved for the SDRAM.

of the PC main memory: It is enough to access the memory, as mapped by the Linux kernel, at the right offset. On the other hand, by using the driver it is possible to access the PCI memory without having to know its absolute offset, thus simplifying further the access to the card. Fig. 4 shows how the memory is mapped on the PCI CFD: A register space is assigned to PCI BAR 5 and is defined as nonprefetchable, a memory space is assigned to BAR 4 and defined as prefetchable. 32 KiB are reserved for the registers, but only 64 B (16 registers, 32 bits each) are used by existing applications. Those regions are mapped in the PLX bridge local memory area and assigned respectively to chip select 0 and 1 [3].

The driver also sets the PC CPU memory type range registers (MTRR) [9]. These are used to set the communication strategy between areas of the memory and the processor cache. They are needed to achieve optimum performance; in fact, a factor 4 to 5 or more in performance can be achieved by simply setting these registers to the proper values.

From the preceding remarks, it should be clear that the interfacing with the software (DAQ and configuration) of a PCI device is extremely simple, as PCI is natively supported by most operating systems.

C. Performance

Good performance was achieved, especially considering that the card is working as a PCI target. This was possible thanks to the appropriate setting of the MTRR registers (see Section III-B) and to the fact that both the PLX and the SDRAM were working with eight-word bursts. The MTRR setting used (*write through*) allows for full CPU cache line fills when reading from the PCI bus [9].

It should be noted that the performance of a PCI device strongly depends on the host computer. The performance that is quoted here was achived on a test system[2] (based on a ASUS A7M266 motherboard), which was carefully chosen in 2001 in order to optimize it. Differences of about 30% between different host computers were observed.

[2]In fact, this is the system which was chosen and is used for the readout of the NA60 experiment.

TABLE I
PIII: INTEL PENTIUM III, 800 MHz. Ath 800: ATHLON, 800 MHz. Ath 1.2: AMD ATHLON, 1.2 GHz

Motherboard	Processor	Reading	Writing
ASUS CUBX	PIII	12 MiB/s	25 MiB/s
MSI 815EP Pro	PIII	12 MiB/s	20 MiB/s
ASUS A7M266	Ath 800	12 MiB/s	28 Mib/s
ASUS A7M266	Ath 1.2	15 MiB/s	30 MiB/s

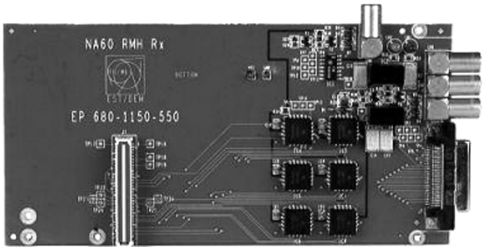


Fig. 5. Mezzanine for level conversion from ECL/NIM signals to TTL. Initially developed for the readout of the NA60 Muon Spectrometer. It is now used in the readout of three different detectors.

The bandwidth for accessing the SDRAM is 30 MiB/s when reading and 15 MiB/s when writing. Without the MTRR setting the observed performance is 5 MiB/s when reading.

Table I summarizes results achieved with different motherboards and processors.

IV. APPLICATIONS

The card is presently used in several different applications, in the readout of the NA60 experiment and in a test readout system for the ALICE experiment muon chambers.

NA60 applications will be described briefly (a detailed description of the NA60 readout system can be found in [2]).

The first set of NA60 applications is based on a very simple mezzanine (Fig. 5) which only does the level conversion from ECL and NIM signals, used by the front-end electronics, to TTL signals which can be used by the FPGA on the CFD. The readout protocol is implemented inside the FPGA itself. This mezzanine was initially developed for the readout of the Muon Spectrometer, but it is now also used for the readout of two other detectors (zero degree calorimeter and beam tracker). This has required implementing two different protocols: the RMH [10] and the FERA [11] protocol. Both of them are double handshake protocols, very similar to each other and similar to the VME one. The main difference between the two is that in the case of the RMH the PCI-CFD card is the bus-master, while in the case of the FERA it is the slave.

The readout of the pixel detector is the most complex application developed for the NA60 experiment. It is based on a mezzanine (Fig. 6) equipped with a FPGA which performs readout, zero suppression, and data encoding. Encoded data are written into a FIFO, which is then accessed by the CFD card. The mezzanine is also responsible for the JTAG configuration of the on-detector electronics and itself has some configurable parameters (e.g., the length and delay of the signal used to strobe the data). Commands are sent to the mezzanine, which executes them or forward them to the front-end electronics, with the mechanism described in Section III-A.

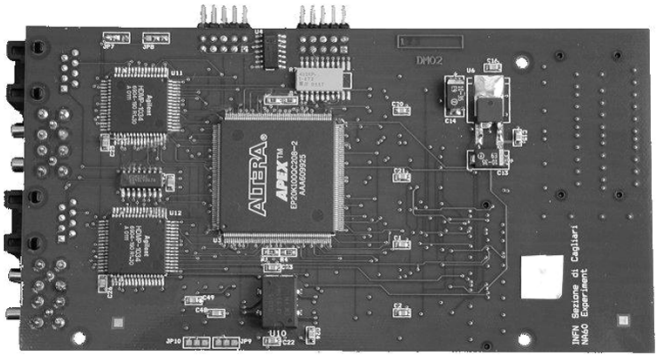


Fig. 6. NA60 pixel readout mezzanine. This complex mezzanine is used to implement the custom NA60 pixel readout system.

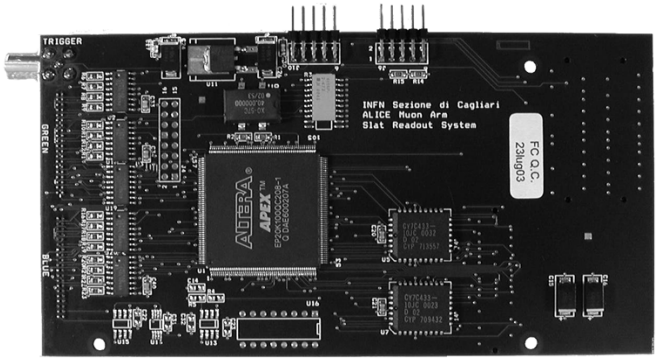


Fig. 7. ALICE mezzanine. This mezzanine is used to implement a readout test system for the ALICE Muon Arm tracking chambers. It implements a DSP linkport emulation.

One application has also been developed out of the scope of NA60. The ALICE experiment [12], which will take place at the CERN LHC, will make use of MWPC tracking chambers for its "Muon Arm." The chambers for the last stations are currently being assembled. A readout system, to be used for tests with cosmic rays for these chambers was developed. It is based on a complex mezzanine (Fig. 7) and the idea is somewhat similar to the NA60 pixel application: The custom readout protocol is implemented on the mezzanine FPGA and data are written into a FIFO. In the ALICE experiment the muon chambers will be read out using a system of DSPs from Analog Devices, using a custom bus called linkport [12]. An emulation of the DSP accessing the linkport has been implemented in the mezzanine. In this application it is as well necessary to send commands to the on-detector electronics and to the mezzanine.

This application is different from the NA60 ones because it is not spill-buffered. For this reason it has a different control logic (see Section III-A): Every event is transferred to disk as soon as it is produced. A simple custom DAQ software has also been developed for this application.

V. Conclusion

In this paper, a PCI card which has been designed for readout applications in high energy physics has been presented. Despite the very short development time of about two months, the card is highly flexible and has good overall performance. This was possible thanks to hardware PCI cores and FPGA technology. The former permits to develop a PCI card without having to care about the PCI bus itself, the latter allows for great flexibility.

Accurate choice of the host computer and careful tuning of the software has also played an important role in improving the performance.

Although better performances could be achieved with more specialized cards, we think one of the major strengths of our card is *generality*, as the card can be used in any commodity PC.

An upgrade which significantly improves the performance of the card without compromising its generality could be achieved (relatively easily) replacing the presently used PLX bridge with another one working as a PCI master.

The card is currently used as the basic readout element in the CERN experiment NA60 and in a test system for the ALICE muon arm tracking chambers. Interfacing to different detectors is ensured by custom mezzanine cards and further applications might be implemented using the same scheme. Among the future plans is the implementation of the SLINK protocol [5].

Acknowledgment

The authors would like to thank P. Randaccio and R. Marzeddu from Cagliari University for their helpful discussions on the PLX bridges.

References

[1] "Study of Prompt Dimuon and Charm Production With Proton and Heavy Ion Beams at the CERN SPS," NA60 Collaboration, SPSC /P.316, 2000.
[2] M. Floris, D. Marras, G. Usai, A. David, P. Rosinsky, and H. Ohnishi, "The NA60 experiment readout architecture," in *Proc. IEEE NPSS 13th Real Time Conf.*, Montreal, QC, Canada, May 18–23, 2003.
[3] PCI 9030 Data Book. PLX Technologies. [Online]. Available: http://www.plxtech.com
[4] IEEE Standard P1386, 2001.
[5] H. C. Van der Bij, R. A. McLaren, O. Boyle, and G. Rubin, "S-LINK, a data link interface specification for the LHC era," in *Proc. 1996 IEEE Nuclear Science Symposium*, Anaheim, CA, USA, Nov. 2–9, 1996.
[6] IEEE Standard P1386.1, 2001.
[7] IEEE Standard 1149.1a, 1993.
[8] Jam Programming & Test Language Specification. Altera Corporation. [Online]. Available: http://www.altera.com
[9] A. David. (2001) Performance Issues in PCI Reading, NA60 Internal Note 2001-3. [Online]. Available: http://cern.ch/adavid
[10] J. B. Lindsay, C. Millerin, J. C. Tarle, H. Verweij, and H. Wendler, "A fast and flexible data acquisition system for multiwire proportional chambers and other detectors," *Nucl. Instrum. Methods*, vol. 156, pp. 329–333, 1978.
[11] LeCroy FERA Series. LeCroy Corporation. [Online]. Available: http://www.lecroy.com
[12] "Technical Design Report," ALICE Collaboration, CERN/LHCC 99-22, sec. 2.4.1.

Appendix C

Burst-by-burst data selection

Depois da colheita há que separar o trigo do joio.
Provérbio Português

This appendix offers further details on the run and burst selection used for the physics analysis reported in this thesis.

Prior to data reconstruction, runs are preselected by scanning the "run logbooks". We only retain long enough runs (more than 10 bursts) which are not flagged as inappropriate for the physics analyses (calibrations, high-voltage scans, zero-field runs, etc), as described in chapter 6.

Each group of runs comprehends reconstructed runs with a given combination of ACM and PT7 magnet polarities, which are, along with detector setup changes, the most important variables to consider when mixing events from different runs. It is important to keep in mind that event mixing is at the heart of the combinatorial and fake matches background subtraction, as explained in chapter 7.

Data selection is done on a burst-by-burst level. For brevity we present only the "database" used to reject runs in Group 1 and bursts in two runs of that group. Its format is very simple: two numbers identify a run and a burst to reject. If the burst number is -10, then the whole run should be rejected.

Plots of the variables used in identifying bad runs/bursts follow the listings. We show the run survey for Group 1 and the burst survey for two runs, for brevity reasons. As can be noted, for the most part run 6454 (page 203) has no reconstructed dimuons, though the last 7 bursts could be salvaged. Run 6455 shows the same symptoms, but only for a few bursts in the middle of the run. The complete list of groups and runs processed in this selection procedure can be found at `http://cern.ch/adavid/thesis/IMR-burst-selection/`.

For each run, averages over its bursts are implied. Except for the measured beam intensity, all variables are averaged over events in a burst. The panels on the left half (left to right, top to bottom) correspond to:

Beam (ZDC) Beam intensity (integrated per burst) as measured by the ZDC.

Rec PC Dimuons Number of reconstructed dimuons in the muon spectrometer.

Rec PC Dimuons per Beam Ratio of the two previous variables.

PCMuon fraction on Top sextants (charge av.) Charge averaged ratio of single muons reconstructed in the muon spectrometer for sextants above and below the beamline.

Dimuon matching rate Fraction of reconstructed dimuons in the muon spectrometer which were successfully matched to tracks from the vertex tracker.

PCMuon fraction on Jura Sextants (μ^+ only) Fraction of reconstructed positive muons in the sextants on the Jura side of the muon spectrometer.

VTTracks per PC Dimuon Number of reconstructed vertex tracker tracks in events with a reconstructed PC dimuon.

VTDimuon $\frac{\pm\pm}{--}$ ratio Ratio of like-sign matched dimuons.

VTVertex X x coordinate of the interaction point as determined by the vertex tracker tracks' vertex.

VTVertex Y *idem* for the y coordinate.

The right half panels show the average hit occupancy per event in the individual pixel planes, adding together hits from all sensors in each plane.

All these variables have important information concerning the potential use of the events in the corresponding runs/bursts, as described in chapter 6.

In order to streamline the scanning procedure, a program automatically evaluates the dispersion of the data and sets reasonable thresholds for rejection (horizontal red lines). These are stricter for some variables, reflecting the importance of that variable in the selection. This program then produces a list of suspicious bursts/runs together with the (coded) reasons for rejection.

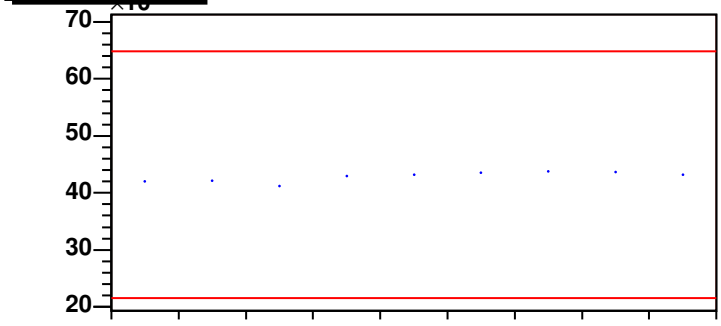
We took these as indications for possible problems and the selection was always done using human discernment. This can be noted from the many lines commented out (starting with the "#" character) as well as bursts inserted by hand (which lack the rejection "reason" after the burst number).

C.1 Group 1 (Runs 6454–6468)

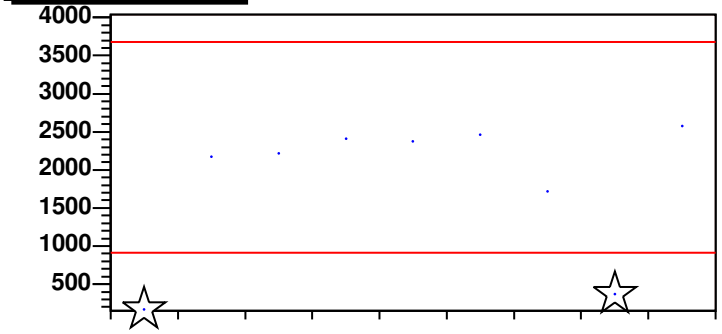
```
Runs suggested for rejection in Group 1 [6454-6468]
# 6454 -10 | Rej: [PCDimu PCDimuPerBeam ]
# 6460 -10 | Rej: [VTVtx ]
# 6464 -10 | Rej: [PCDimuPerBeam ]
 6467 -10 | Rej: [PCDimu PCDimuPerBeam VTVtx ]
# 6468 -10 | Rej: [VTVtx ]

Bursts suggested for rejection in Run 6454 [from 101 bursts]
 6454      1 | Rej: [PCDimu MatchRate TracksPerPCDimu SextAsymm VTVtx LSRatio ]
 6454      2 | Rej: [PCDimu MatchRate TracksPerPCDimu SextAsymm LSRatio ]
 6454      3 | Rej: [PCDimu MatchRate TracksPerPCDimu SextAsymm VTVtx LSRatio ]
 6454      4 | Rej: [PCDimu MatchRate TracksPerPCDimu SextAsymm LSRatio ]
 6454      5 | Rej: [PCDimu MatchRate TracksPerPCDimu SextAsymm VTVtx LSRatio ]
 6454      6 | Rej: [PCDimu MatchRate TracksPerPCDimu SextAsymm VTVtx LSRatio ]
 6454      7 | Rej: [PCDimu MatchRate TracksPerPCDimu SextAsymm LSRatio ]
 6454      8 | Rej: [PCDimu MatchRate TracksPerPCDimu SextAsymm LSRatio ]
 6454      9 | Rej: [PCDimu MatchRate TracksPerPCDimu SextAsymm LSRatio ]
 6454     10 | Rej: [PCDimu MatchRate TracksPerPCDimu SextAsymm LSRatio ]
 6454     11 | Rej: [PCDimu MatchRate TracksPerPCDimu SextAsymm LSRatio ]
 6454     12 | Rej: [PCDimu MatchRate TracksPerPCDimu SextAsymm LSRatio ]
 6454     13 | Rej: [PCDimu MatchRate TracksPerPCDimu SextAsymm LSRatio ]
 6454     14 | Rej: [PCDimu MatchRate TracksPerPCDimu SextAsymm LSRatio ]
 6454     15 | Rej: [PCDimu MatchRate TracksPerPCDimu SextAsymm LSRatio ]
 6454     16 | Rej: [PCDimu MatchRate TracksPerPCDimu SextAsymm VTVtx LSRatio ]
 6454     17 | Rej: [PCDimu MatchRate TracksPerPCDimu SextAsymm LSRatio ]
 6454     18 | Rej: [PCDimu MatchRate TracksPerPCDimu SextAsymm VTVtx LSRatio ]
 6454     19 | Rej: [PCDimu MatchRate TracksPerPCDimu SextAsymm VTVtx LSRatio ]
 6454     20 | Rej: [PCDimu MatchRate TracksPerPCDimu SextAsymm LSRatio ]
 6454     21 | Rej: [PCDimu MatchRate TracksPerPCDimu SextAsymm LSRatio ]
 6454     22 | Rej: [PCDimu MatchRate TracksPerPCDimu SextAsymm VTVtx LSRatio ]
 6454     23 | Rej: [PCDimu MatchRate TracksPerPCDimu SextAsymm LSRatio ]
 6454     24 | Rej: [PCDimu MatchRate TracksPerPCDimu SextAsymm LSRatio ]
 6454     25 | Rej: [PCDimu MatchRate TracksPerPCDimu SextAsymm LSRatio ]
 6454     26 | Rej: [PCDimu MatchRate TracksPerPCDimu SextAsymm VTVtx LSRatio ]
 6454     27 | Rej: [PCDimu MatchRate TracksPerPCDimu SextAsymm LSRatio ]
 6454     28 | Rej: [PCDimu MatchRate TracksPerPCDimu SextAsymm VTVtx LSRatio ]
 6454     29 | Rej: [PCDimu MatchRate TracksPerPCDimu SextAsymm LSRatio ]
 6454     30 | Rej: [PCDimu MatchRate TracksPerPCDimu SextAsymm LSRatio ]
 6454     31 | Rej: [PCDimu MatchRate TracksPerPCDimu SextAsymm LSRatio ]
 6454     32 | Rej: [PCDimu MatchRate TracksPerPCDimu SextAsymm LSRatio ]
 6454     33 | Rej: [PCDimu MatchRate TracksPerPCDimu SextAsymm LSRatio ]
 6454     34 | Rej: [PCDimu MatchRate TracksPerPCDimu SextAsymm LSRatio ]
 6454     35 | Rej: [PCDimu MatchRate TracksPerPCDimu SextAsymm LSRatio ]
 6454     36 | Rej: [PCDimu MatchRate TracksPerPCDimu SextAsymm LSRatio ]
 6454     37 | Rej: [PCDimu MatchRate TracksPerPCDimu SextAsymm LSRatio ]
 6454     38 | Rej: [PCDimu MatchRate TracksPerPCDimu SextAsymm LSRatio ]
 6454     39 | Rej: [PCDimu MatchRate TracksPerPCDimu SextAsymm LSRatio ]
 6454     40 | Rej: [PCDimu MatchRate TracksPerPCDimu SextAsymm LSRatio ]
 6454     41 | Rej: [PCDimu MatchRate TracksPerPCDimu SextAsymm LSRatio ]
 6454     42 | Rej: [PCDimu MatchRate TracksPerPCDimu SextAsymm VTVtx LSRatio ]
 6454     43 | Rej: [PCDimu MatchRate TracksPerPCDimu SextAsymm LSRatio ]
 6454     44 | Rej: [PCDimu MatchRate TracksPerPCDimu SextAsymm VTVtx LSRatio ]
 6454     45 | Rej: [PCDimu MatchRate TracksPerPCDimu SextAsymm VTVtx LSRatio ]
 6454     46 | Rej: [PCDimu MatchRate TracksPerPCDimu SextAsymm VTVtx LSRatio ]
 6454     47 | Rej: [PCDimu MatchRate TracksPerPCDimu SextAsymm LSRatio ]
 6454     48 | Rej: [PCDimu MatchRate TracksPerPCDimu SextAsymm LSRatio ]
 6454     49 | Rej: [PCDimu MatchRate TracksPerPCDimu SextAsymm LSRatio ]
 6454     50 | Rej: [PCDimu MatchRate TracksPerPCDimu SextAsymm LSRatio ]
 6454     51 | Rej: [PCDimu MatchRate TracksPerPCDimu SextAsymm LSRatio ]
 6454     52 | Rej: [PCDimu MatchRate TracksPerPCDimu SextAsymm LSRatio ]
 6454     53 | Rej: [PCDimu MatchRate TracksPerPCDimu SextAsymm LSRatio ]
 6454     54 | Rej: [PCDimu MatchRate TracksPerPCDimu SextAsymm LSRatio ]
 6454     55 | Rej: [PCDimu MatchRate TracksPerPCDimu SextAsymm VTVtx LSRatio ]
 6454     56 | Rej: [PCDimu MatchRate TracksPerPCDimu SextAsymm VTVtx LSRatio ]
 6454     57 | Rej: [PCDimu MatchRate TracksPerPCDimu SextAsymm LSRatio ]
 6454     58 | Rej: [PCDimu MatchRate TracksPerPCDimu SextAsymm LSRatio ]
 6454     59 | Rej: [PCDimu MatchRate TracksPerPCDimu SextAsymm LSRatio ]
 6454     60 | Rej: [PCDimu MatchRate TracksPerPCDimu SextAsymm VTVtx LSRatio ]
 6454     61 | Rej: [PCDimu MatchRate TracksPerPCDimu SextAsymm LSRatio ]
 6454     62 | Rej: [PCDimu MatchRate TracksPerPCDimu SextAsymm LSRatio ]
 6454     63 | Rej: [PCDimu MatchRate TracksPerPCDimu SextAsymm LSRatio ]
 6454     64 | Rej: [PCDimu MatchRate TracksPerPCDimu SextAsymm LSRatio ]
 6454     65 | Rej: [PCDimu MatchRate TracksPerPCDimu SextAsymm VTVtx LSRatio ]
 6454     66 | Rej: [PCDimu MatchRate TracksPerPCDimu SextAsymm VTVtx LSRatio ]
 6454     67 | Rej: [PCDimu MatchRate TracksPerPCDimu SextAsymm LSRatio ]
 6454     68 | Rej: [PCDimu MatchRate TracksPerPCDimu SextAsymm VTVtx LSRatio ]
 6454     69 | Rej: [PCDimu MatchRate TracksPerPCDimu SextAsymm LSRatio ]
 6454     70 | Rej: [PCDimu MatchRate TracksPerPCDimu SextAsymm LSRatio ]
 6454     71 | Rej: [PCDimu MatchRate TracksPerPCDimu SextAsymm LSRatio ]
 6454     72 | Rej: [PCDimu MatchRate TracksPerPCDimu SextAsymm LSRatio ]
 6454     73 | Rej: [PCDimu MatchRate TracksPerPCDimu SextAsymm LSRatio ]
 6454     74 | Rej: [PCDimu MatchRate TracksPerPCDimu SextAsymm LSRatio ]
 6454     75 | Rej: [PCDimu MatchRate TracksPerPCDimu SextAsymm LSRatio ]
 6454     76 | Rej: [PCDimu MatchRate TracksPerPCDimu SextAsymm LSRatio ]
 6454     77 | Rej: [PCDimu MatchRate TracksPerPCDimu SextAsymm VTVtx LSRatio ]
 6454     78 | Rej: [PCDimu MatchRate TracksPerPCDimu SextAsymm LSRatio ]
 6454     79 | Rej: [PCDimu MatchRate TracksPerPCDimu SextAsymm LSRatio ]
 6454     80 | Rej: [PCDimu MatchRate TracksPerPCDimu SextAsymm LSRatio ]
 6454     81 | Rej: [PCDimu MatchRate TracksPerPCDimu SextAsymm LSRatio ]
 6454     82 | Rej: [PCDimu MatchRate TracksPerPCDimu SextAsymm VTVtx LSRatio ]
 6454     83 | Rej: [PCDimu MatchRate TracksPerPCDimu SextAsymm LSRatio ]
 6454     84 | Rej: [PCDimu MatchRate TracksPerPCDimu SextAsymm LSRatio ]
 6454     85 | Rej: [PCDimu MatchRate TracksPerPCDimu SextAsymm LSRatio ]
 6454     86 | Rej: [PCDimu MatchRate TracksPerPCDimu SextAsymm VTVtx LSRatio ]
 6454     87 | Rej: [PCDimu MatchRate TracksPerPCDimu SextAsymm LSRatio ]
 6454     88 | Rej: [PCDimu MatchRate TracksPerPCDimu SextAsymm VTVtx LSRatio ]
 6454     89 | Rej: [PCDimu MatchRate TracksPerPCDimu SextAsymm LSRatio ]
 6454     90 | Rej: [PCDimu MatchRate TracksPerPCDimu SextAsymm LSRatio ]
 6454     91 | Rej: [PCDimu MatchRate TracksPerPCDimu SextAsymm LSRatio ]
 6454     92 | Rej: [PCDimu MatchRate TracksPerPCDimu SextAsymm VTVtx LSRatio ]
 6454     93 | Rej: [PCDimu MatchRate TracksPerPCDimu SextAsymm LSRatio ]
 6454     94 | Rej: [PCDimu MatchRate TracksPerPCDimu SextAsymm LSRatio ]

Bursts suggested for rejection in Run 6455 [from 94 bursts]
 6455     33 | Rej: [PCDimu MatchRate TracksPerPCDimu SextAsymm LSRatio ]
 6455     34 | Rej: [PCDimu MatchRate TracksPerPCDimu SextAsymm VTVtx LSRatio ]
 6455     35 | Rej: [PCDimu MatchRate TracksPerPCDimu SextAsymm VTVtx LSRatio ]
 6455     36 | Rej: [PCDimu MatchRate TracksPerPCDimu SextAsymm LSRatio ]
 6455     37 | Rej: [PCDimu MatchRate TracksPerPCDimu SextAsymm VTVtx LSRatio ]
 6455     38 | Rej: [PCDimu MatchRate TracksPerPCDimu SextAsymm VTVtx LSRatio ]
 6455     39 | Rej: [PCDimu MatchRate TracksPerPCDimu SextAsymm LSRatio ]
 6455     40 | Rej: [PCDimu MatchRate TracksPerPCDimu SextAsymm VTVtx LSRatio ]
 6455     41 | Rej: [PCDimu MatchRate TracksPerPCDimu SextAsymm VTVtx LSRatio ]
 6455     42 | Rej: [PCDimuPerBeam VTVtx ]
 6455     47 | Rej: [PCDimu MatchRate TracksPerPCDimu SextAsymm VTVtx LSRatio ]
 6455     94 | Rej: [PCDimuPerBeam VTVtx ]
Bursts suggested for rejection in Run 6457 [from 27 bursts]
# 6457      2 | Rej: [VTVtx ]
# 6457      5 | Rej: [VTVtx ]
 6457      9 | Rej: [PCDimu PCDimuPerBeam TracksPerPCDimu VTVtx ]
 6457     18 | Rej: [PCDimu PCDimuPerBeam TracksPerPCDimu VTVtx ]

Bursts suggested for rejection in Run 6458 [from 86 bursts]
# 6458      3 | Rej: [VTVtx ]
```

```
# 6458    13 | Rej: [VTVtx ]
  6458    16 | Rej: [PCDimu PCDimuPerBeam VTVtx ]
# 6458    21 | Rej: [VTVtx ]
# 6458    25 | Rej: [VTVtx ]
# 6458    27 | Rej: [VTVtx ]
# 6458    28 | Rej: [VTVtx ]
# 6458    29 | Rej: [VTVtx ]
# 6458    30 | Rej: [VTVtx ]
# 6458    35 | Rej: [VTVtx ]
# 6458    37 | Rej: [VTVtx ]
# 6458    40 | Rej: [VTVtx ]
# 6458    42 | Rej: [VTVtx ]
# 6458    46 | Rej: [VTVtx ]
# 6458    48 | Rej: [VTVtx ]
# 6458    49 | Rej: [VTVtx ]
# 6458    50 | Rej: [VTVtx ]
# 6458    51 | Rej: [VTVtx ]
# 6458    53 | Rej: [VTVtx ]
# 6458    54 | Rej: [VTVtx ]
# 6458    57 | Rej: [VTVtx ]
# 6458    59 | Rej: [VTVtx ]
# 6458    64 | Rej: [VTVtx ]
# 6458    70 | Rej: [VTVtx ]
# 6458    71 | Rej: [VTVtx ]
# 6458    72 | Rej: [VTVtx ]
# 6458    73 | Rej: [VTVtx ]
  6458    74 | Rej: [PCDimuPerBeam VTVtx ]
# 6458    80 | Rej: [VTVtx ]
# 6458    81 | Rej: [VTVtx ]
# 6458    82 | Rej: [VTVtx ]
  6458    86 | Rej: [PCDimuPerBeam VTVtx ]

Bursts suggested for rejection in Run 6460 [from 89 bursts]
  6460     3 | Rej: [PCDimuPerBeam VTVtx ]
  6460    36 | Rej: [PCDimuPerBeam VTVtx ]
  6460    55 | Rej: [PCDimu PCDimuPerBeam VTVtx ]
  6460    60 | Rej: [PCDimu PCDimuPerBeam VTVtx ]
  6460    89 | Rej: [PCDimu PCDimuPerBeam TracksPerPCDimu VTVtx ]

Bursts suggested for rejection in Run 6462 [from 17 bursts]
  6462     1 | Rej: [VTVtx ]
# 6462     9 | Rej: [VTVtx ]
  6462    14 | Rej: [VTVtx ]
  6462    16 | Rej: [VTVtx ]
  6462    17 | Rej: [PCDimuPerBeam ]

Bursts suggested for rejection in Run 6464 [from 92 bursts]
  6464    52 | Rej: [PCDimuPerBeam VTVtx ]
  6464    69 | Rej: [VTVtx ]
  6464    70 | Rej: [VTVtx ]
  6464    80 | Rej: [PCDimu PCDimuPerBeam TracksPerPCDimu VTVtx ]


Bursts suggested for rejection in Run 6468 [from 60 bursts]
  6468    49 | Rej: [PCDimuPerBeam ]
```

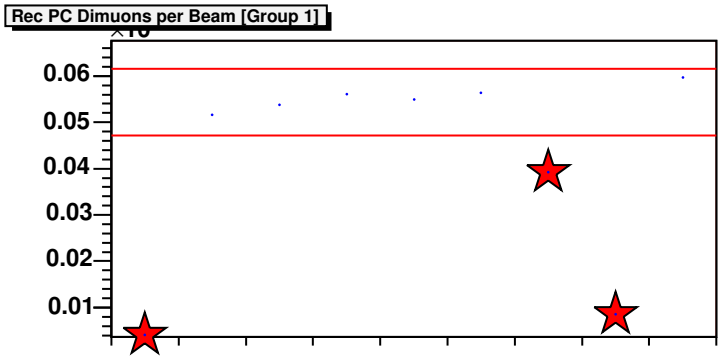
Beam (ZDC) [Group 1]



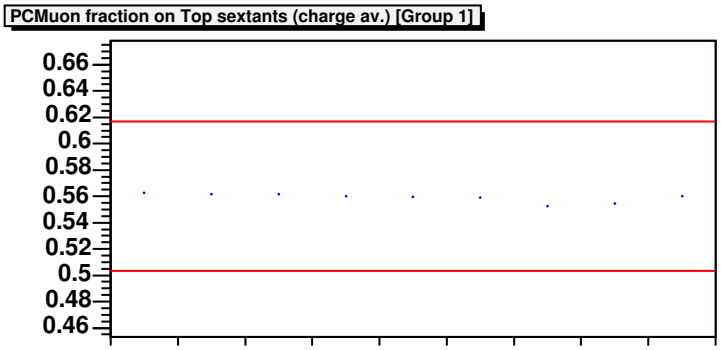
Rec PC Dimuons [Group 1]



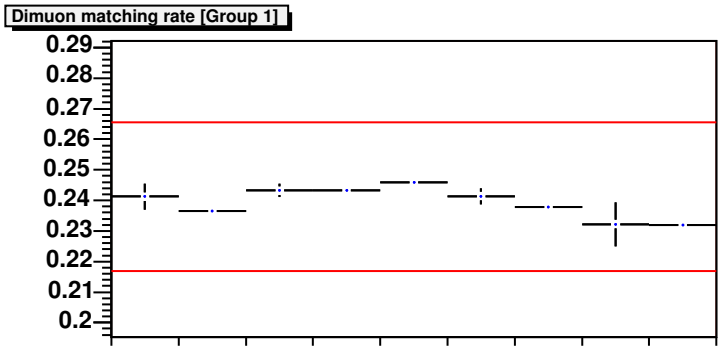
Rec PC Dimuons per Beam [Group 1]



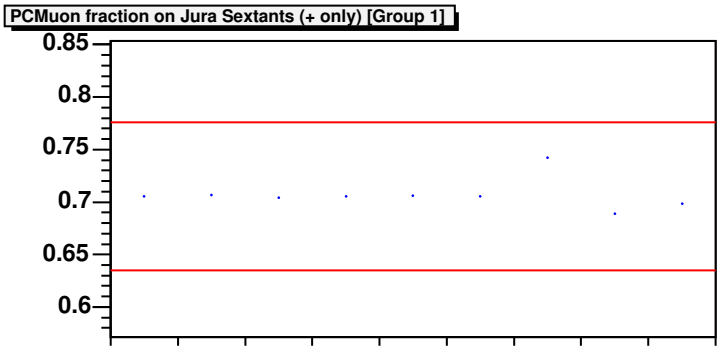
PCMuon fraction on Top sextants (charge av.) [Group 1]



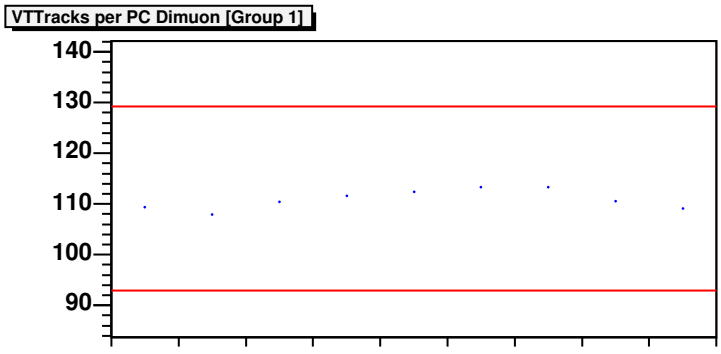
Dimuon matching rate [Group 1]



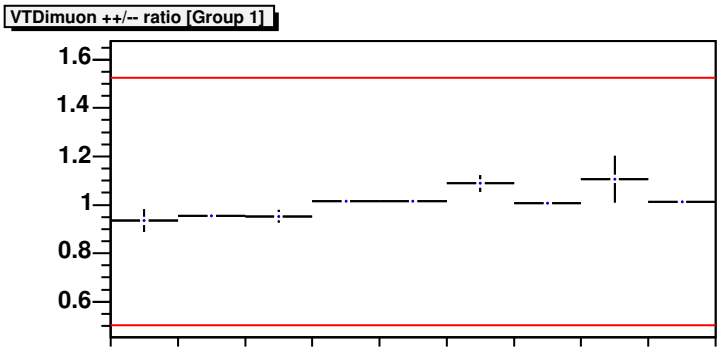
PCMuon fraction on Jura Sextants (+ only) [Group 1]



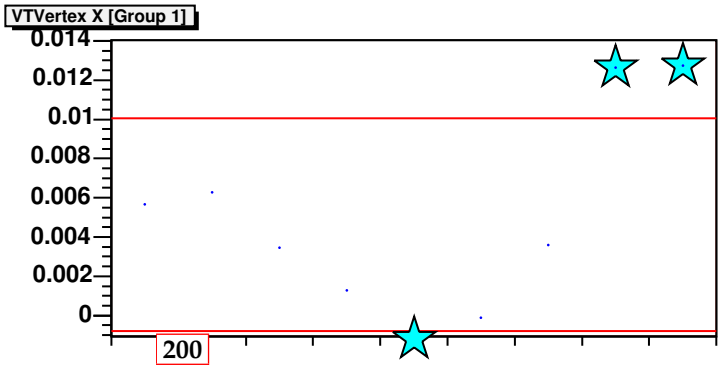
VTTracks per PC Dimuon [Group 1]



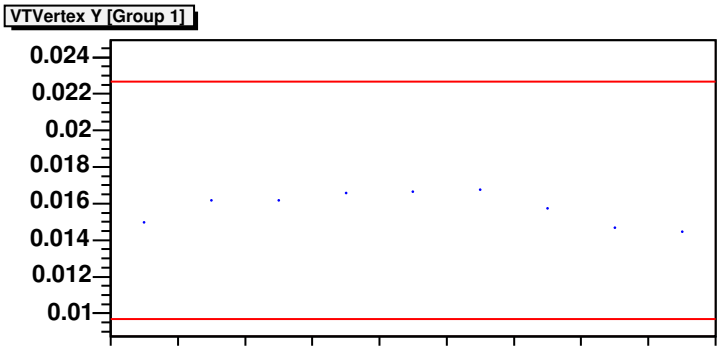
VTDimuon ++/-- ratio [Group 1]

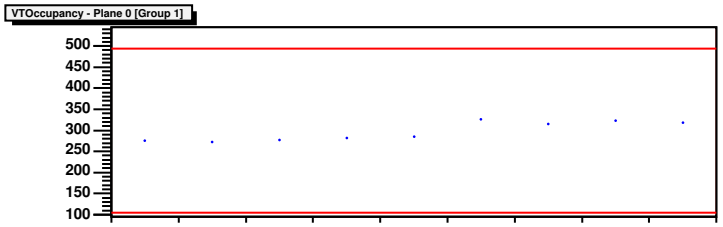


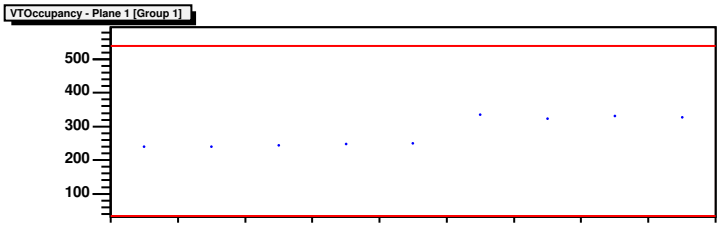
VTVertex X [Group 1]

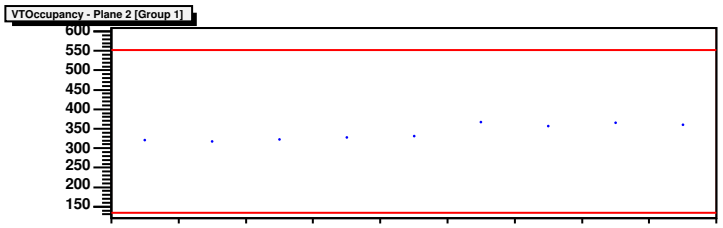


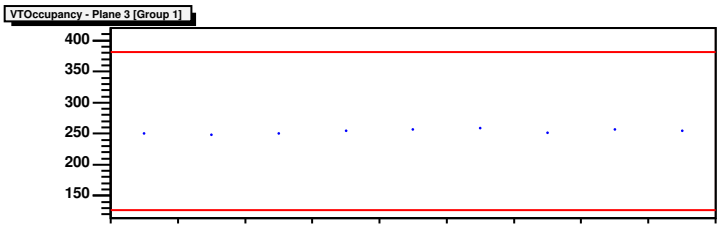
VTVertex Y [Group 1]

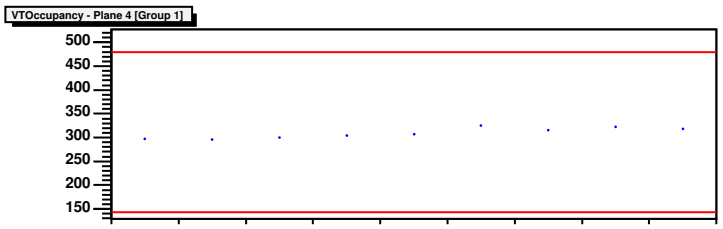


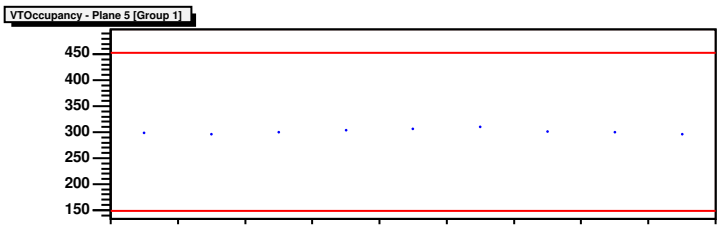

VTOccupancy - Plane 0 [Group 1]
500
450
400
350
300
250
200
150
100

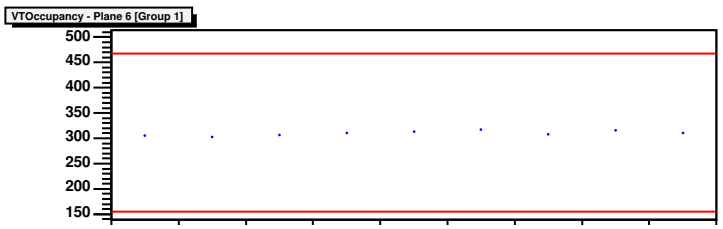

VTOccupancy - Plane 1 [Group 1]
500
400
300
200
100

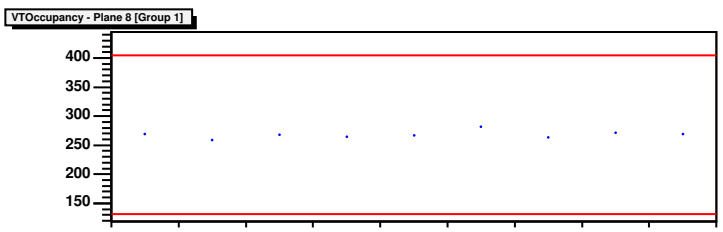

VTOccupancy - Plane 2 [Group 1]
600
550
500
450
400
350
300
250
200
150

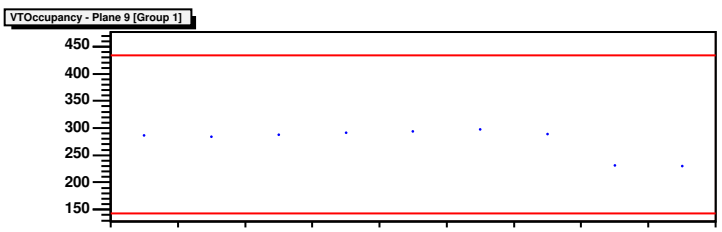

VTOccupancy - Plane 3 [Group 1]
400
350
300
250
200
150

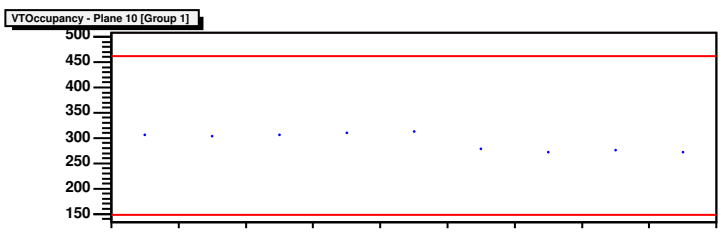

VTOccupancy - Plane 4 [Group 1]
500
450
400
350
300
250
200
150

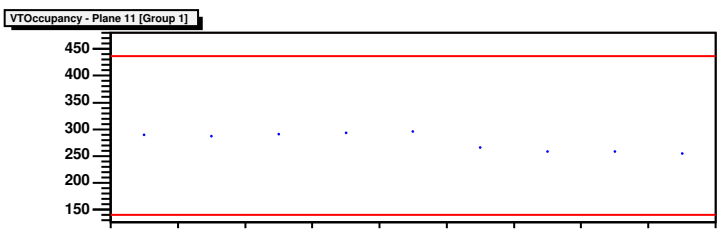

VTOccupancy - Plane 5 [Group 1]
450
400
350
300
250
200
150

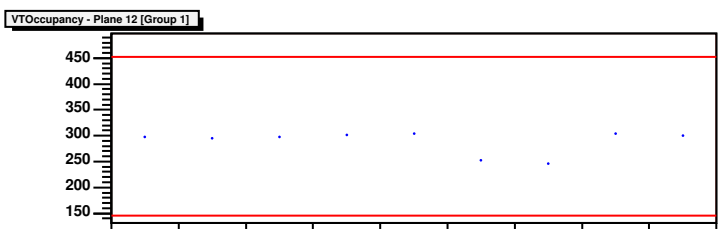

VTOccupancy - Plane 6 [Group 1]
500
450
400
350
300
250
200
150

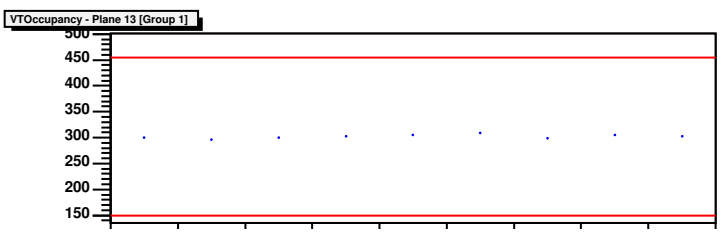

VTOccupancy - Plane 8 [Group 1]
400
350
300
250
200
150

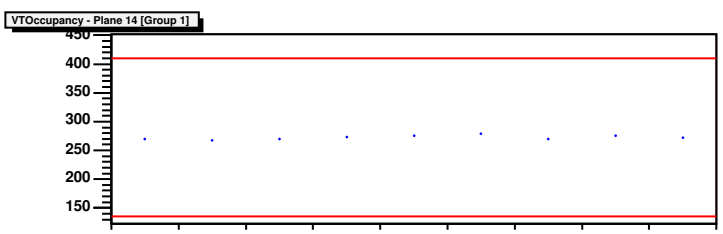

VTOccupancy - Plane 9 [Group 1]
450
400
350
300
250
200
150

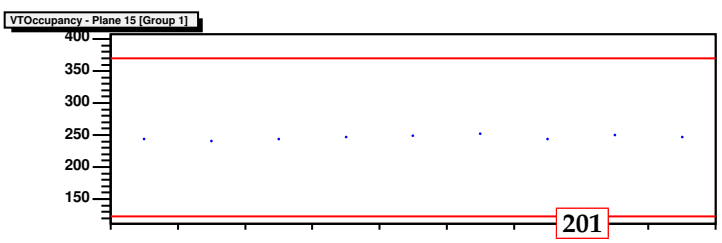

VTOccupancy - Plane 10 [Group 1]
500
450
400
350
300
250
200
150


VTOccupancy - Plane 11 [Group 1]
450
400
350
300
250
200
150

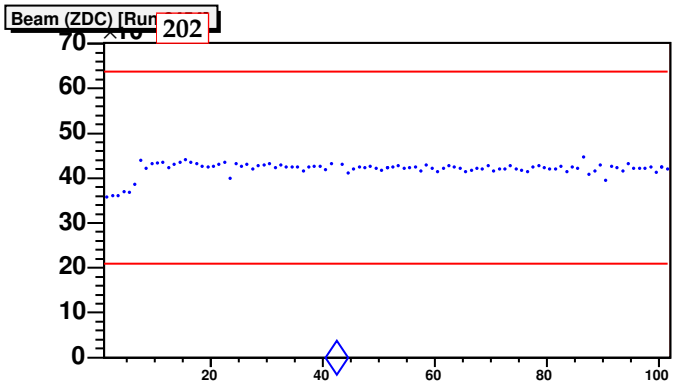

VTOccupancy - Plane 12 [Group 1]
450
400
350
300
250
200
150


VTOccupancy - Plane 13 [Group 1]
500
450
400
350
300
250
200
150

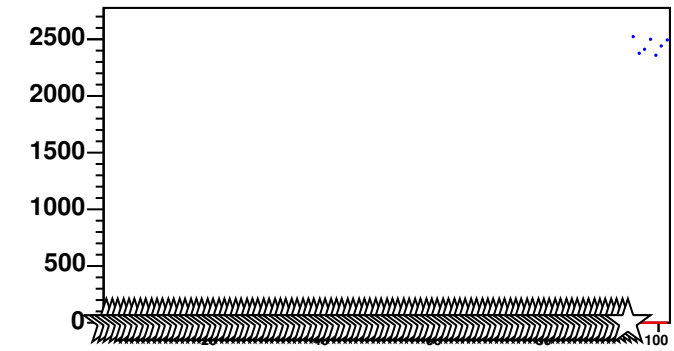

VTOccupancy - Plane 14 [Group 1]
450
400
350
300
250
200
150


VTOccupancy - Plane 15 [Group 1]
400
350
300
250
200
150

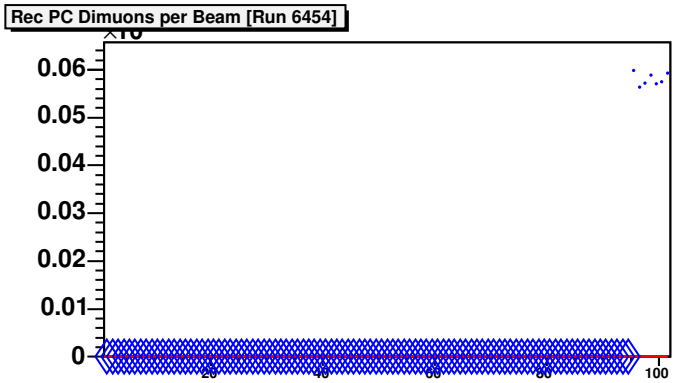
Beam (ZDC) [Run 202]



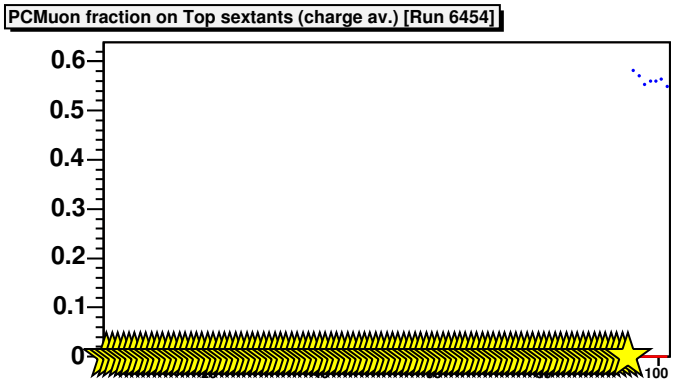
Rec PC Dimuons [Run 6454]



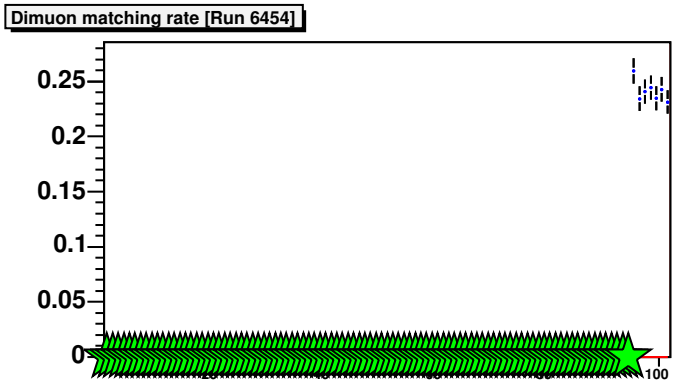
Rec PC Dimuons per Beam [Run 6454]



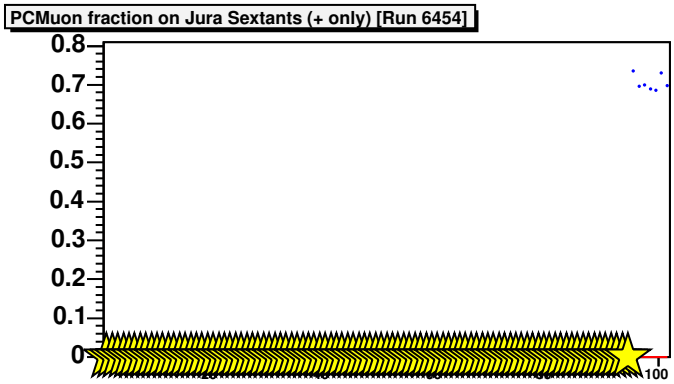
PCMuon fraction on Top sextants (charge av.) [Run 6454]



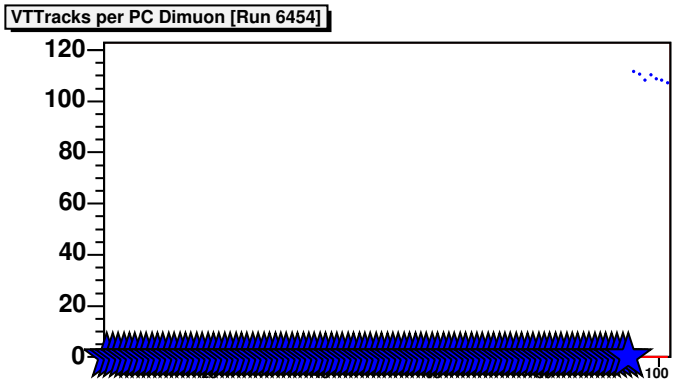
Dimuon matching rate [Run 6454]



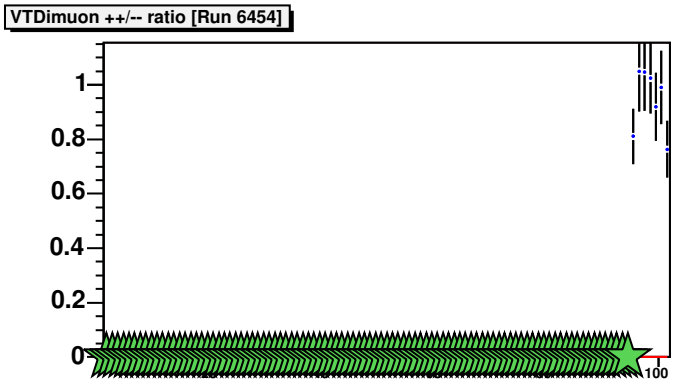
PCMuon fraction on Jura Sextants (+ only) [Run 6454]



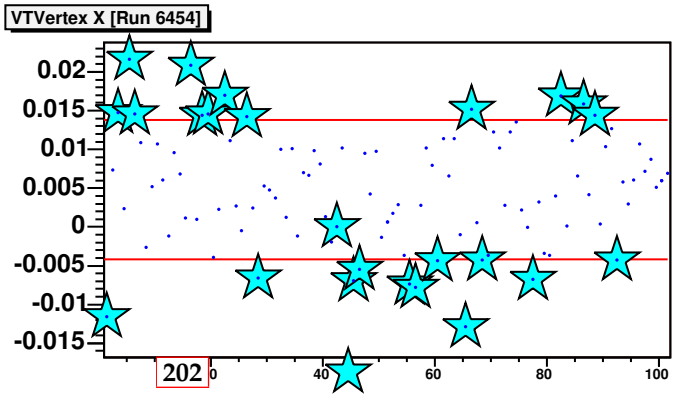
VTTracks per PC Dimuon [Run 6454]



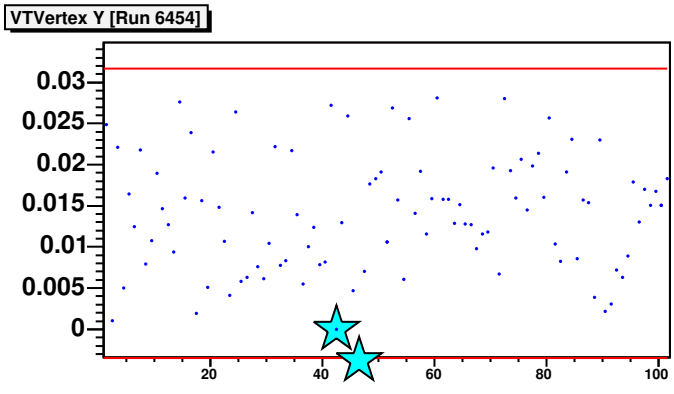
VTDimuon ++/-- ratio [Run 6454]

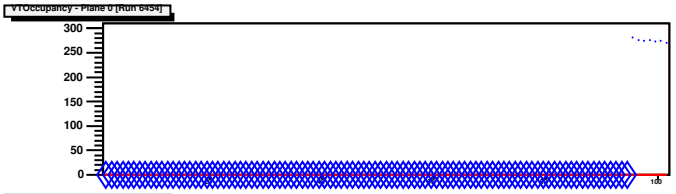


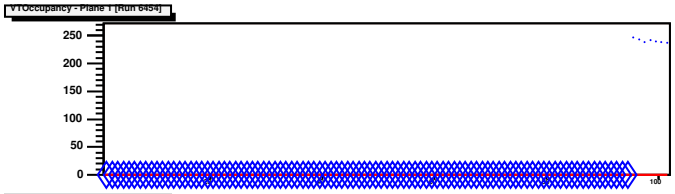
VTVertex X [Run 6454]

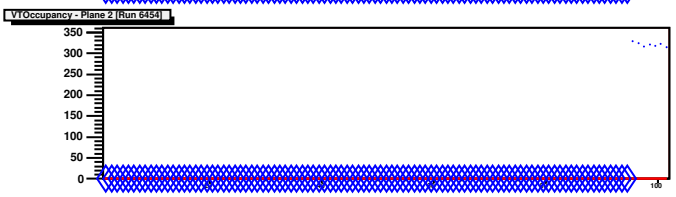


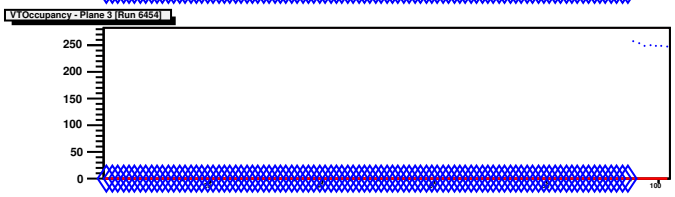
VTVertex Y [Run 6454]

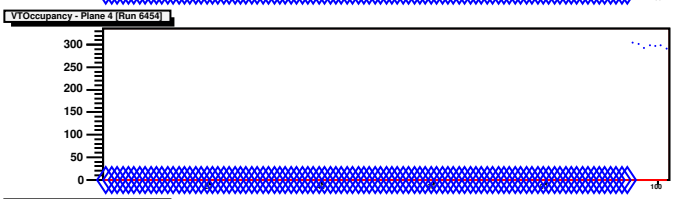


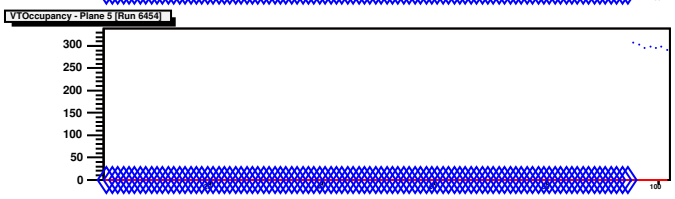

VTOccupancy - Plane 0 [Run 6454]
300
250
200
150
100
50
0
100

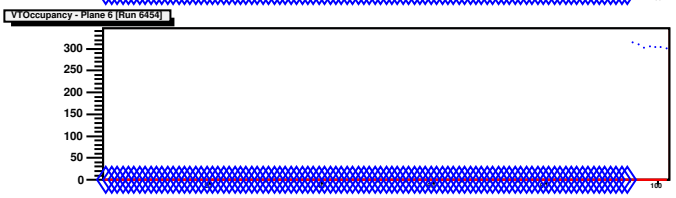

VTOccupancy - Plane 1 [Run 6454]
250
200
150
100
50
0
100

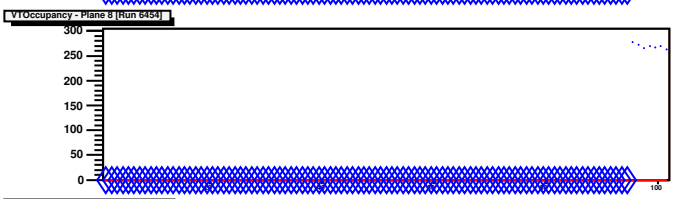

VTOccupancy - Plane 2 [Run 6454]
350
300
250
200
150
100
50
0
100

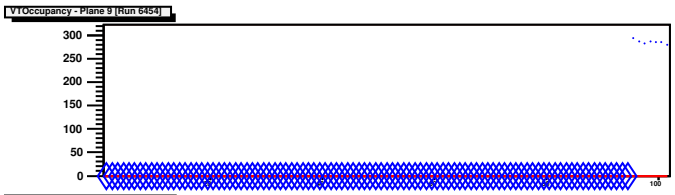

VTOccupancy - Plane 3 [Run 6454]
250
200
150
100
50
0
100

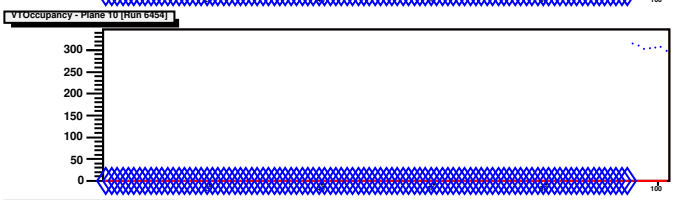

VTOccupancy - Plane 4 [Run 6454]
300
250
200
150
100
50
0
100

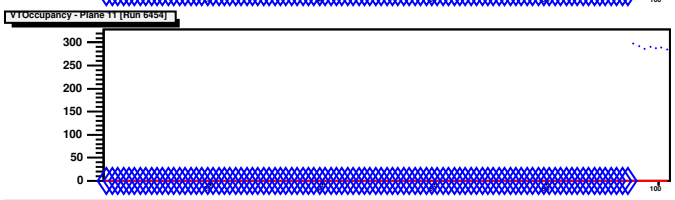

VTOccupancy - Plane 5 [Run 6454]
300
250
200
150
100
50
0
100


VTOccupancy - Plane 6 [Run 6454]
300
250
200
150
100
50
0
100


VTOccupancy - Plane 8 [Run 6454]
300
250
200
150
100
50
0
100

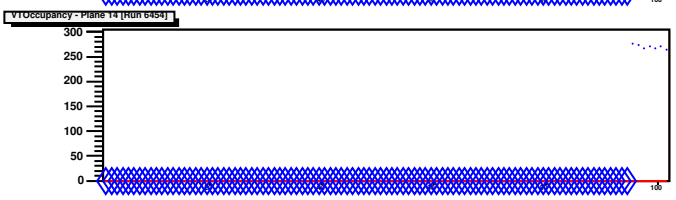

VTOccupancy - Plane 9 [Run 6454]
300
250
200
150
100
50
0
100

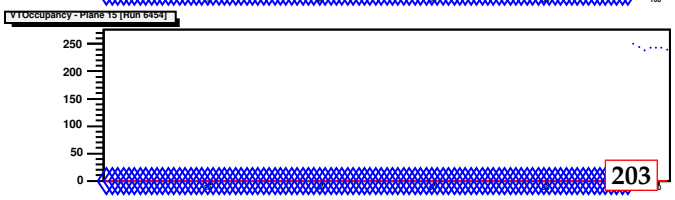

VTOccupancy - Plane 10 [Run 6454]
300
250
200
150
100
50
0
100


VTOccupancy - Plane 11 [Run 6454]
300
250
200
150
100
50
0
100


VTOccupancy - Plane 12 [Run 6454]
300
250
200
150
100
50
0
100


VTOccupancy - Plane 13 [Run 6454]
300
250
200
150
100
50
0
100


VTOccupancy - Plane 14 [Run 6454]
300
250
200
150
100
50
0
100


VTOccupancy - Plane 15 [Run 6454]
250
200
150
100
50
0
100