# Determining Shaper Signal Characteristics From TileCal Testbeam Data

Ben Cowan

May 5, 1999

## 1 Introduction

In this document we describe a method for finding the shape of the signal from the TileCal 3-in-1 cards in a realistic ATLAS environment using testbeam data. We show how to extract the relevant information from a testbeam Ntuple and reconstruct the signal shape, and then we use some reconstructed shapes to examine the behavior of the signal width over a range of energies deposited in a channel and to compare signal widths from electrons, muons, and pions in the beam.

The procedure for reconstructing signal shapes involves several steps:

1. Use TILEMON to produce an Ntuple for a testbeam run with raw ADC counts rather than computed energies, and use a "preprocessing" program to cut down this enormous Ntuple to a manageable size.
2. Make cuts to select the desired type of particles and then choose a range of estimated energies of signals to look at.
3. Plot the signals and weed out oddly behaved events.

In the next several sections we describe this procedure in detail. We illustrate the methods using a specific example, run 43162, which has a beam of $50$ GeV electrons at $-90$ degrees through tile 6. The electrons in this run deposit energy in cell B$-9$, which is in sample depth 2 of the module at $\eta = -0.85$. In the course of the description we mention pieces of code that were used to perform the analysis. This code can be accessed on the web at `http://hep.uchicago.edu/~benc/`.

## 2 Ntuple Production

To look at signal shapes we must produce an Ntuple with TILEMON that contains the raw ADC samples. To do this, we set the `FERNT` flag in the `tmon.datacards` file to `-1`. Since the Ntuple produced is quite large, we need to set the `MKNTU` card to `4` and increase the size of TILEMON's common block. We also need to modify the TILEMON code to include the high/low gain flag. The procedure for making these changes is documented in January 8 and 18, 1999 posts to the tileanal news archive by Bob Stanek. Note that as of the last TILEMON release a gain flag of 0 corresponds to low gain and 1 to high gain. It is also helpful, to save storage space and processing time, to omit non-physics events from the Ntuple by setting the cards `UPEDE` and `ULASE` to `0`. Finally, in the `COLNT` datacard, turn off all columns except 12, 14, 15, and 16; the preprocessing program described in the next paragraph depends on this.

The Ntuple that TILEMON produces is quite large, over 70 megabytes, too cumbersome to analyze efficiently with PAW. We therefore use a small FORTRAN "preprocessing" program, called TILEPP, to create a much smaller Ntuple (about one megabyte) which contains only the samples for our channel of interest,

as well as some other useful information. The TILEPP program takes two command-line arguments, the first being the name of the Ntuple produced by TILEMON and the second being a name for the new Ntuple. In our example we would type:

```
tilepp r0043162.hbook r0043162a.hbook
```

to produce a new Ntuple with filename `r0043162a.hbook`.

The columns of the new Ntuple are as follows:

- `event`: This is the event tag, identical to the one in the TILEMON Ntuple, except that it is stored as an integer.
- `trig`: The trigger word, identical to the one in the TILEMON Ntuple, except stored as an integer.
- `TDC`: This is the TDC value, divided by 20 to be in units of nanoseconds.
- `samples(40)`: These are the 40 samples in the channel of interest.
- `pedestal`: The mean of the first 10 samples in the signal.
- `integral`: This is an uncalibrated estimate of the energy deposited in the channel of interest. It is computed by finding the largest sample between the 10th and 34th inclusive, adding that sample, the two just before it, and the two just after, and then subtracting the value of `pedestal`.
- `emuon`: This variable is used as a muon discriminator. In the case of our example, in which the beam is incident at $-90$ degrees, `emuon` is the sum of the uncalibrated estimated energies in the ten channels in cells BC$-1$ through BC$-5$, with the energy in each channel estimated by the same method used to compute `integral`.
- `gain`: This is the gain flag for the channel of interest.

The channel of interest is hardwired into the code in the file `tilepp.f`. To change it, one needs to modify the lines of code where `samples` and `gain` are assigned. (The assignments of `pedestal` and `integral` don't need to be changed because the computations use the `samples` array.) Also, one needs to determine a good muon discriminator for the run being used and assign it to the `emuon` variable; for fixed $\eta$ runs with pions the `mubacksum` variable from the TILEMON Ntuple works well. To recompile the code, one also needs the `aux.C` file and the Makefile.

## 3 Choosing an Energy Range

Now that we have a nice trim preprocessed Ntuple, we are ready to begin hacking away at the cuts to select signals. Since TILEMON's rejection of non-physics events isn't always guaranteed, we first establish the cut `trig` $= 1$. In our example we are concerned with the high-gain channel, so we set `gain` $= 1$. We want to look as signals from electron events, so we set `emuon` $< 100$, as this forms a good muon rejection condition.

Since we want to look at signals that are similar, we should choose a small interval of estimated energies for the signals we plot. On the other hand, with a wider interval we have more events so the signal may be better resolved. We can overcome this trade-off to some extent by plotting not the actual signals, but the signals divided by their estimated energies. This will give us good resolution of the signal shape, although we lose the information about the absolute amplitude. These issues will become clearer when we describe plotting the signals in the next section.

As our interval of estimated energies we choose the peak of the distribution so that we get good statistics with the smallest possible interval. The distribution of the `integral` variable is shown in Figure 1; the shaded area is the interval $[3000, 4000)$ of estimated energies we have chosen. The "Entries" statistic is the number of events with estimated energies in this interval.

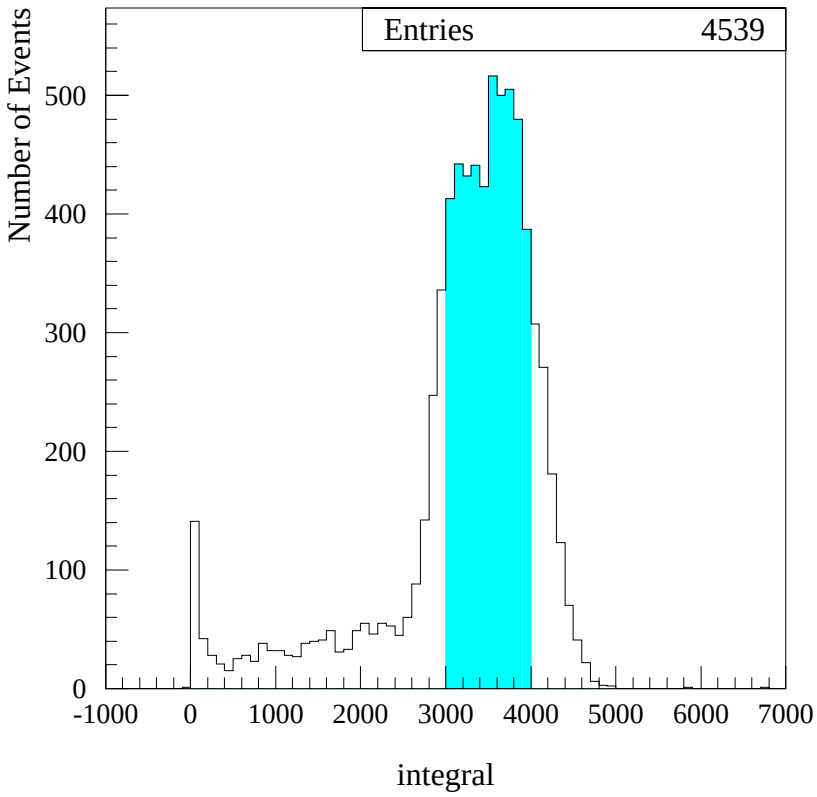


Figure 1: The distribution of estimated energies in a channel.

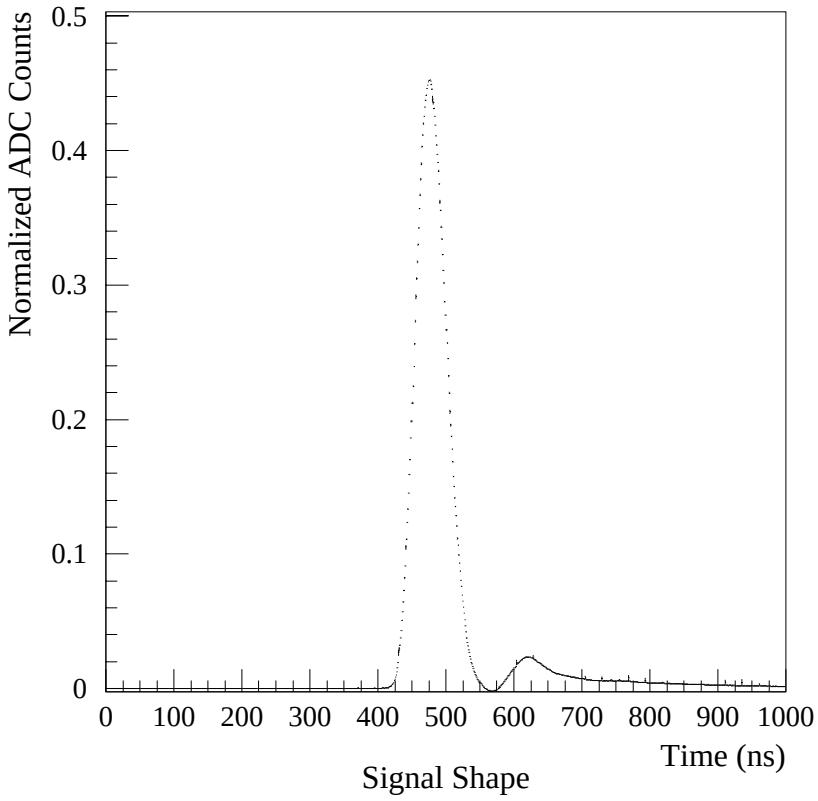


Figure 2: A signal shape plotted as a profile histogram.

## 4 Plotting and Event Weeding

When graphing a set of signals we plot, for each sample in each event, the strength of the signal at that sample versus the time of the sample. We take the time of the sample to be measured from the time that the particle in the beam triggers the TDC. Thus, up to some multiple of $25$ ns which is constant over the entire run, the time of sample $i$ in a signal is

$$t_i = 25i + \texttt{TDC}.$$

We take the strength of the signal at sample $i$ to be

$$x_i = \frac{\texttt{samples}(i) - \texttt{pedestal}}{\texttt{integral}}.$$

There are two ways to plot the signals. The first way uses a profile histogram with 1000 bins, with the time ranging from $0$ ns to $1000$ ns. When we plot $x_i$ versus $t_i$ for each sample $i$ in each event that meets our cuts, the value of each bin will be the mean of the values of $x_i$ for which $t_i$ falls in the bin; the error bars represent the error in the distribution of the $x_i$ in that bin. Such a plot is shown in Figure 2. The plot was created using a PAW macro in the file `shape.kumac`. The macro takes three arguments, the lower bound of our estimated energy interval, the upper bound of the interval, and the type of plot. The type of plot must be either 1 or 2; for plots with profile histograms it's 1, and for the other type, to be discussed later, the type argument is 2. So to make the plot shown in Figure 2, we type

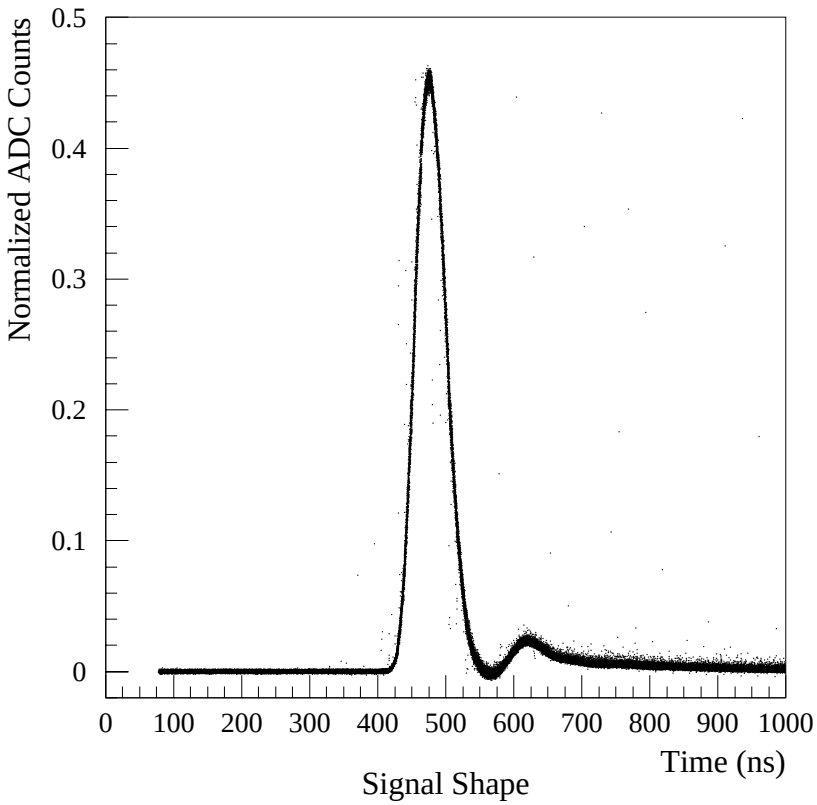


Figure 3: Signals plotted with each sample as a separate point. There are 4539 events included in this plot.

```
macro/exec shape.kumac 3000 4000 1
```

at the PAW command line. The macro creates the histogram (with ID 100), fills it, plots it, and adds axis labels. Note that it also changes the tick spacing on the $x$-axis to $25\ \mathrm{ns}$. This type of plot is just what we're after, as it shows the shape of the signal, resolved in time intervals of $1\ \mathrm{ns}$, with an error on each interval. Now the reason for dividing by the estimated energy becomes clear: The estimated energies in our interval vary by more than 10%. Assuming the amplitude of the signal varies by that much as well, the errors in each interval would be much larger.

However, there are some pathologies with the plot shown in Figure 2. The error bars at certain discrete points are much larger than the almost invisible errors in most of the signal. To examine these problems, we can look at the signals in greater detail using a 2-dimensional histogram, with each sample in each event appearing as a single point. To do so, we execute the macro again, typing

```
macro/exec shape.kumac 3000 4000 2
```

at the PAW command line. The plot is shown in Figure 3. Those points in the plot outside the main band of signals are not just stray flecks of toner from the printer; they are signals which are offset from the others. In some events, two pulses appear in the same signal window, and in others there is just one pulse which is offset considerably in time from the bulk of the events.

We need to cut these misbehaving events out of our sample. To eliminate the events with pulses in the

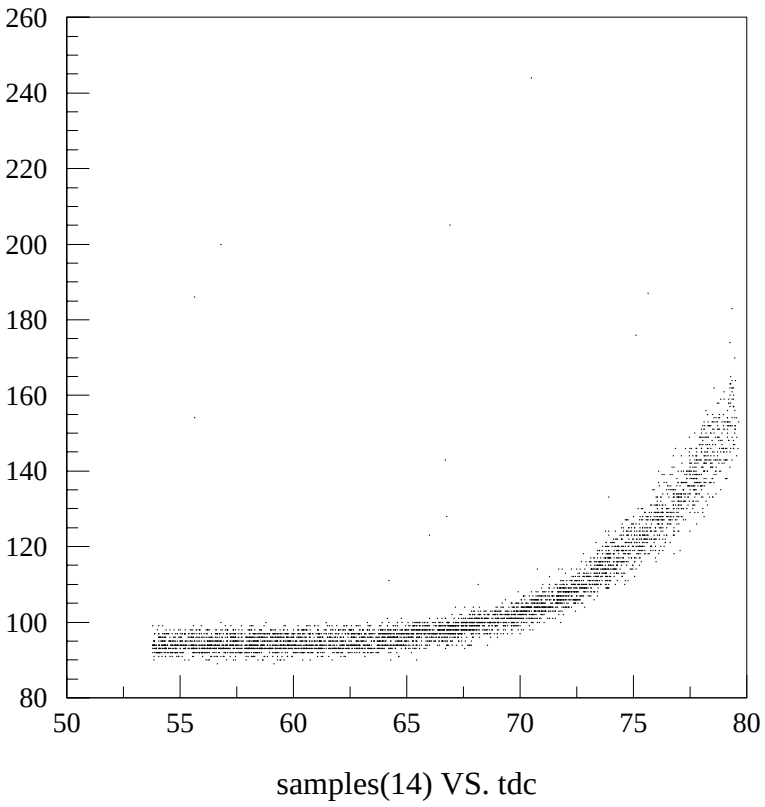


Figure 4: A $25$ ns interval of the signals plot.

tail, we define a PAW selection function called `pedtail.f`, which is defined by

$$\texttt{pedtail} = \sum_{i=22}^{40} \texttt{samples}(i).$$

Setting the cut $\texttt{pedtail.f} < 2500$ cuts out the events with pulses in the tail.

Cutting out the pulses which are merely offset in time is a bit trickier. It's a boring and tedious process, but if you've read this far in this paper, that shouldn't bother you. To isolate these events, we plot `samples(`$i$`)` versus `TDC` for some $i$ that is likely to contain samples in the pulse, for all events that have passed the cuts mentioned so far. This will show a $25$ ns interval of the 2-dimensional plot in greater detail. For instance, `samples(14)` is plotted versus `TDC` in Figure 4. From this plot we see that we can eliminate some events just by cutting on `samples(14)`. We can isolate other events by cutting on both `TDC` and `samples(14)`, and then use the `ntuple/scan` command to find the event numbers of pathological events. To introduce these cuts when plotting the signal shape, we need to edit the `shape.kumac` file. We can then make the 2-dimensional plot again, and then try to cut out more events by repeating this process with different values of $i$ until all the misbehaving signals have been eliminated. Weeding out events this way may seem like cheating; after all, if this timing misbehavior occurs, won't we have to take that into account when we apply this analysis? However, in ATLAS the timing of the pulses in a channel will be fixed with respect to the digitizing clock, since the position of each cell will be fixed with respect to the interaction point. Therefore, pulses will not be offset in time. Pile-up will be a problem, but that is a separate issue which is beyond the scope of this analysis.

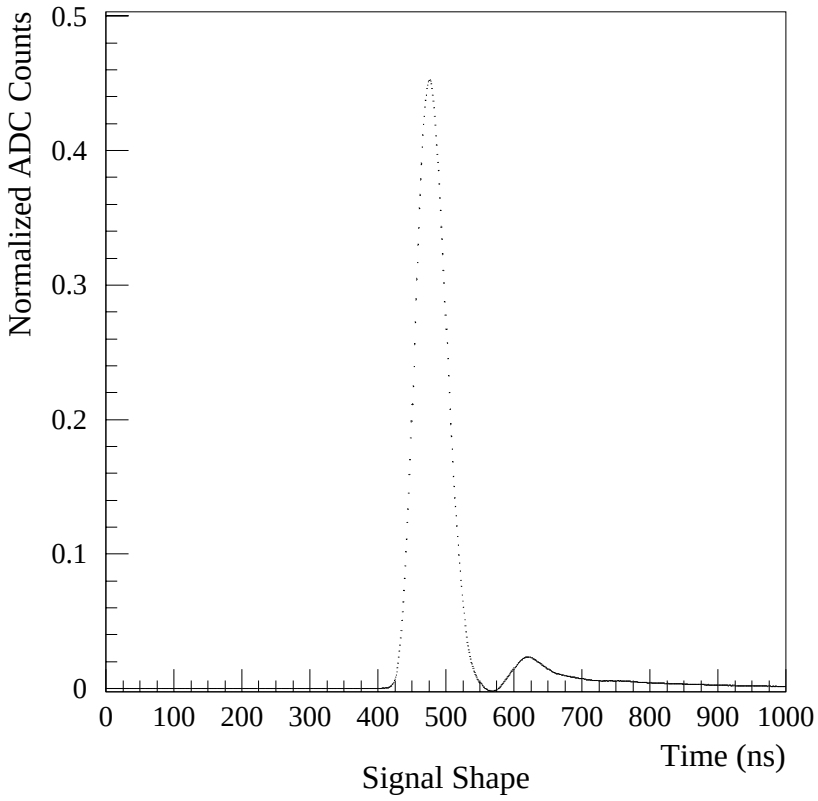


Figure 5: Signals plotted after misbehaving events have been weeded out.

In our example, we wind up with the cuts `pedtail.f` $< 2500$ and `samples(14)` $< 170$. We also cut out individual events using the `event` variable in the Ntuple; the events we reject are 2519, 3341, 5610, 6802, 9514, 6093, and 6945. Of the 4539 events in our estimated energy interval, a total of 21 events have been rejected. Not all runs have this many bad events, but this was useful to illustrate the procedure. A plot of the profile histogram using these cuts appears in Figure 5.

# 5 Signal Widths

Now that we have a method for obtaining signal shapes from testbeam data, we use some shapes to compute signal characteristics for several different beams over a range of energies. We wish to compare the widths of signals resulting from beams of electrons, muons, and pions. Ideally, we should make this comparison for a single channel. However, we wish to examine pion signals in a channel in the second sample depth, as signal pathologies in pion data are most likely to appear there; and we must use a channel in a cell on the outside of the module as this is where electrons deposit energy. The only option would be to examine pion runs at $|\eta| = 0.75$ or $0.85$, and for the 1998 test beam period there was not a broad enough selection of these runs. Therefore, we use electron runs at $-90$ degrees through tile 6 and examine signals in a channel of cell B$-9$, in sample depth 2 at $\eta = -0.85$. We use pion runs at $\eta = -0.35$ and examine signals in a channel of cell BC$-4$ in sample depth 2. We use muon data in both channels to compare signals between the channels.

For each run we compute the full width at half maximum and the full width at 15% of maximum from our

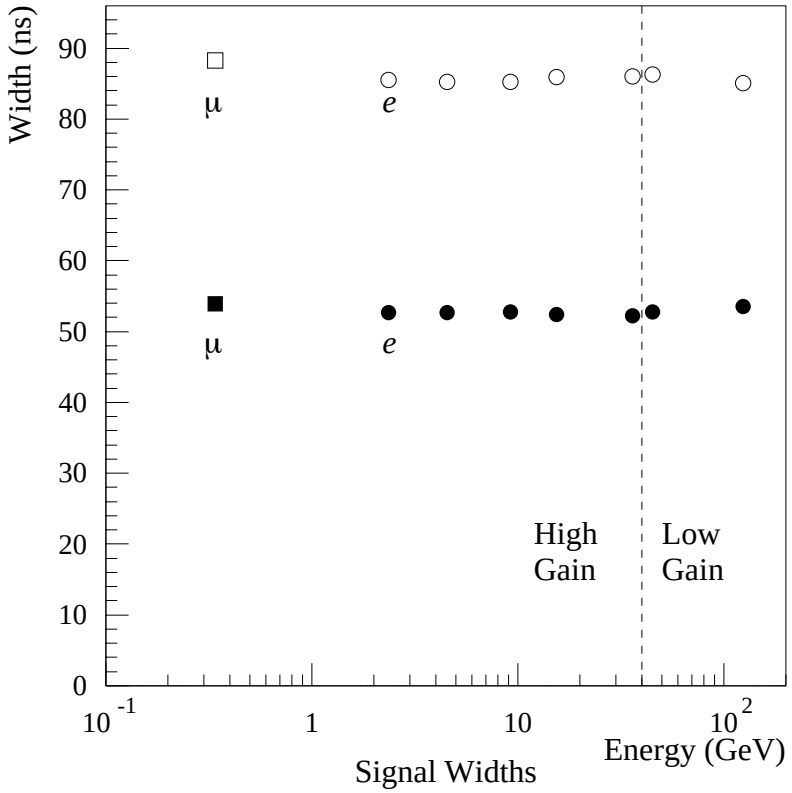


Figure 6: Signal widths in a channel of cell B−9. The horizontal axis represents the amount of energy deposited in the channel, not the total energy of the beam. Filled markers denote the FWHM, and empty markers denote the full width at 15%; circles represent electron data and squares muon data. Also, points to the right of the dotted line are for low gain, the others for high gain.

reconstructed signals. The macro `shape.kumac` calls the FORTRAN function `width.f`, which computes the widths with errors. Figures 6 and 7 show the widths of signals from the two channels. Electrons with energies from $5$ GeV to $300$ GeV and pions with energies from $10$ GeV to $400$ GeV were used. The muon data from channel 7 was taken from a run of $10$ GeV pions, and the muon data from channel 17 was taken from a $180$ GeV muon run. In general, the errors on the widths are a few tenths of a nanosecond, so the error bars are covered by the points on the graph.

# 6 Conclusion

We have systematically used our procedure for reconstructing signal shapes to determine the widths of signals from test beam data. We found that the widths are essentially constant over a range of energies, with the FWHM's varying by no more than $2.3$ ns (4.6%) in a single category.

Reconstructing signals from test beam data has additional applications. We can examine segments the signals around the peak and possibly use this information to develop an algorithm for determining the peak signal amplitude using timing information. Work along these lines is in progress.

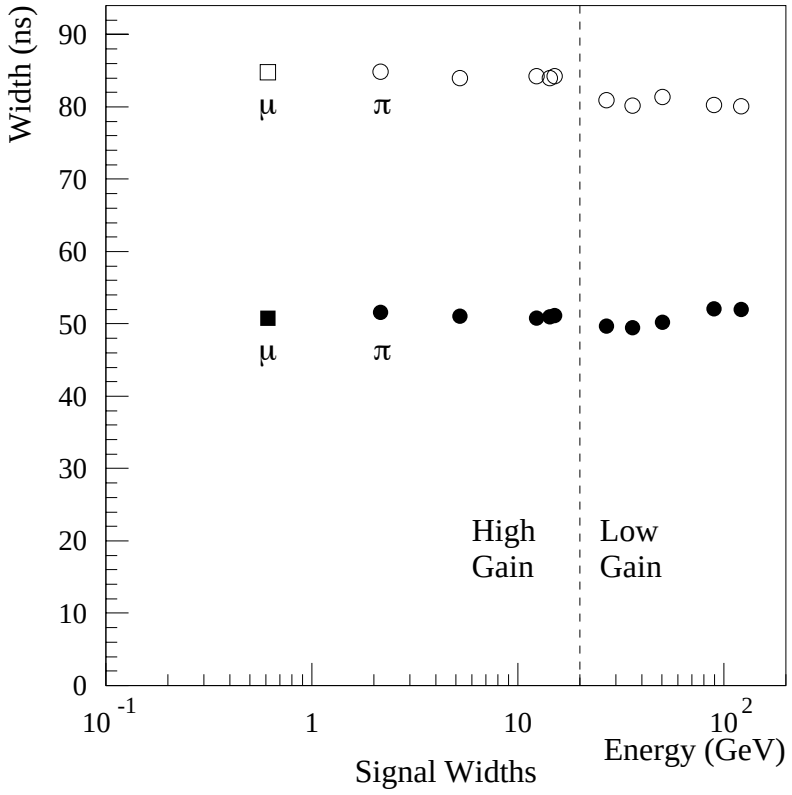


Figure 7: Signal widths in a channel of cell BC$-4$. The horizontal axis represents the amount of energy deposited in the channel, not the total energy of the beam. Filled markers denote the FWHM, and empty markers denote the full width at 15%; circles represent pion data and squares muon data. Also, points to the right of the dotted line are for low gain, the others for high gain.