MS, LG 04/06/2007

Single-hit detection efficiency for a cluster-finder algorithm implemented in the readout chain of the HFT

Goals:

The goal of this study was to estimate the single-hit detection efficiency and fake-hit rate as a function of different threshold criteria for two different cluster finding algorithms that could be used in the HFT readout system. To estimate these characteristics a set of simulations was performed using real data generated with MimoStar2 prototype pixel sensors.

Cluster finder algorithm:

A commonly used cluster-finding algorithm is based on two cuts on signal-to-noise ratio (S/N): the first cut is on the central pixel and the second cut on the sum of S/N of all 8 neighbors (often referred to as a crown). Typical values are 5 and 2 for the central pixel and the crown, respectively. It provides very good detection efficiency and effectively filters accidentals caused by noise. This algorithm can easily be implemented in software. Implementation in hardware – either on a signal processing FPGA or on-chip – is also possible but would require significant resources. In addition to the required multi-bit adder blocks, one also needs a large memory to contain the average noise value for each pixel.
A simpler algorithm was proposed for the implementation in the readout system of the HFT. The idea is to use only two thresholds. A cluster would be recognized when the signal in a central pixel passes a high threshold and, at the same time, one of the eight neighbors passes the second, lower threshold.

In this study the proposed algorithm is investigated and its efficiency and fake-hit rate are considered as a function of the cuts applied to pixel signals.
The result is compared to an algorithm where the second cut is performed on a sum of signals in all 8 neighboring pixels.

Data used for simulations:

The simulations were carried out using real data obtained by the IPHC group with a MimoStar2 prototype. Data came from beam tests that were carried out in summer 2006 at DESY, Germany, with a prototype running at temperature of 20 C and the integration time of 2.1 ms. The data used for the simulation consisted of 1897 clusters (size of 5x5 pixels)[1] extracted as hits in the MimoStar2 prototype that were associated with a particle path reconstructed from reference detectors. Another part of data was 4193 regions of 5x5 pixels containing noise signal extracted from a reference run taken without a beam and in the same running conditions. All signals in clusters were presented in e- with a conversion rate of 7.1 e-/ADC. For the simulation purposes all data was converted into

[1] Charge generated by an impinging particle is shared between several adjacent pixels. Cluster size of 5x5 pixels is large enough to enclose all pixels that collected charge.

ADC units. Standard deviation for noise samples was 2.14 ADC. The most probable value of signal in the central pixel of a cluster was about 30 ADC.

The data input into simulations contained clusters embedded into background signal. Events with a frame size of 15x15 pixels were formed from noise data with the central 5x5 region replaced by a single cluster. To allow study of fake-hit rate down to rates of 10^{-6}, the initial 104825 noise samples were used for generating 1.2 M noise samples for the simulation. A pseudo-random number generator randomly selected samples from the initial set of data and built the large set used for the simulation. The distribution of the generated set of data was in agreement with the distribution of the original noise samples. The complete set of data used for the study consisted of 7588 frames with a size of 15x15 pixels (each signal cluster was used 4 times)

Definitions:
Detection efficiency:

$$eff = \frac{N_{org}}{N_{TOT}} \times 100 = \frac{N_{org}}{N_{FRAMES}} \times 100 \ [\%]$$

N_{org}– number of clusters detected in the original central 5x5 regions of all entry events,
N_{TOT} – the total number of clusters embedded in the central regions of entry events equal to the number of frames (N_{FRAMES}=7588).

Fake-hit rate per pixel:

$$FHR = \frac{N_F}{N_{FRAMES} \times N_{PIX}} = \frac{\sum_{frames}\left(n_{\det} - n_{org}\right)}{N_{FRAMES} \times N_{PIX}}$$

N_F – number of fake clusters,
n_{det}– number of clusters detected in one frame,
n_{org}– number of clusters detected in one frame at the position where the original cluster was embedded,
N_{PIX} =160 – number of pixels in one frame that are scanned for clusters, excluding the original cluster of 25 pixels (in addition, two rows at the edges of each frame where not scanned).

Procedure:
Each frame was scanned with a 3x3 pixel window. The central pixel is considered as the center of a cluster when the cluster criterion is met. As a result a number of pixels can be marked in one cluster. An example for this procedure is presented in Figure 1.

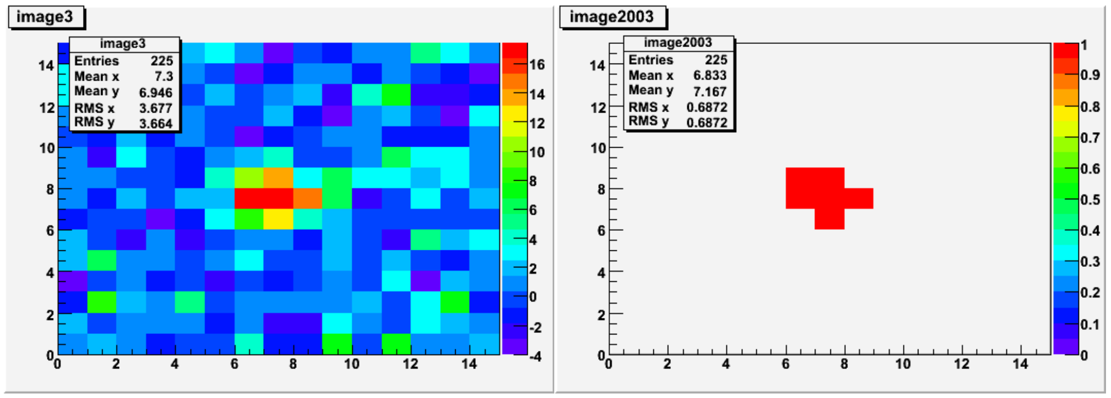

Figure 1 Images of the original frame and the frame after passing through a cluster finding algorithm.

Reconstruction of a cluster from a group of pixels was accomplished using a flood-fill algorithm. The binary image was raster scanned and when a valid pixel was encountered a recursive function was called to merge pixels into a single cluster. The centroid of each cluster has to be found to estimate the hit position. A very similar algorithm will be required for processing data from the HFT.

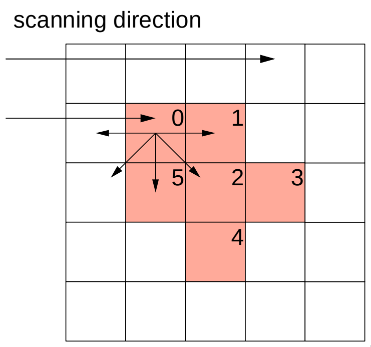

Figure 2 Flood-fill algorithm that merges neighboring pixels into a single cluster.

The distance between the hit position estimated from a center of mass of the original cluster and the centroid of a reconstructed cluster has the distribution presented in Figure 3. The mean value is compatible with a binary resolution that is equal to

$$\sigma = \frac{p}{\sqrt{12}}$$

where p is pixel pitch that, in this case, is normalized to 1.

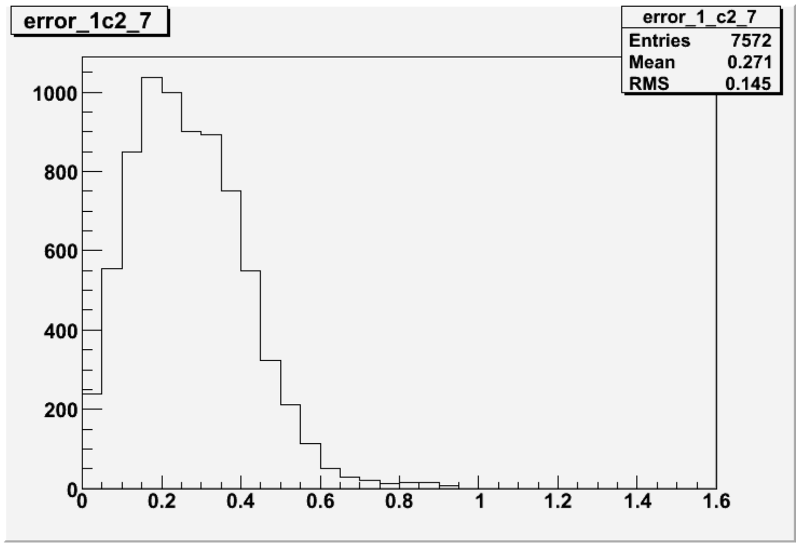

Figure 3 Distance between a center of mass for the original cluster (12 bit resolution) and the centroid of the reconstructed cluster.

The size of a cluster depends on the high cut on the central pixel. Average cluster size as a function of a cut on the central pixel is presented in Figure 4 and it varies from about 4 to 2.6 pixels for lower and higher cuts respectively.

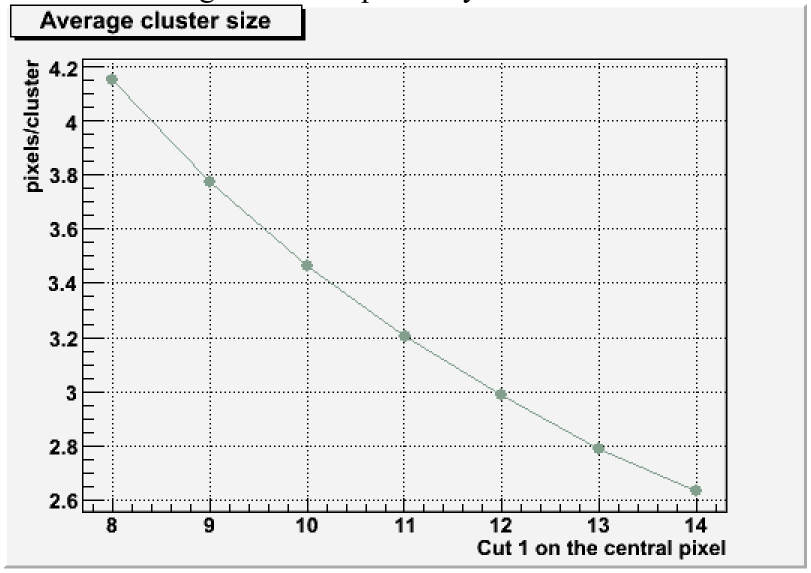

Figure 4 Average number of pixels in a cluster as a function of the high cut on the central pixel in the cluster finding algorithm.

Results:

Figure 5 shows efficiency vs. fake-hit rate per pixel for the proposed two-threshold cluster finding algorithm as a function of applied cuts expressed in ADC counts.
Figure 6 shows the same characteristics for the algorithm where the second cut is performed on a sum of signals in 8 adjacent pixels.

Different colors show different settings for the second cut, while the points on the curves correspond to different threshold on the central pixel that vary from 14 to 8 ADC units with a step of one ADC unit when tracked from left to right.
The fake-hit rate equal to 10^{-7} is an arbitrary number that corresponds to no fake hits detected.

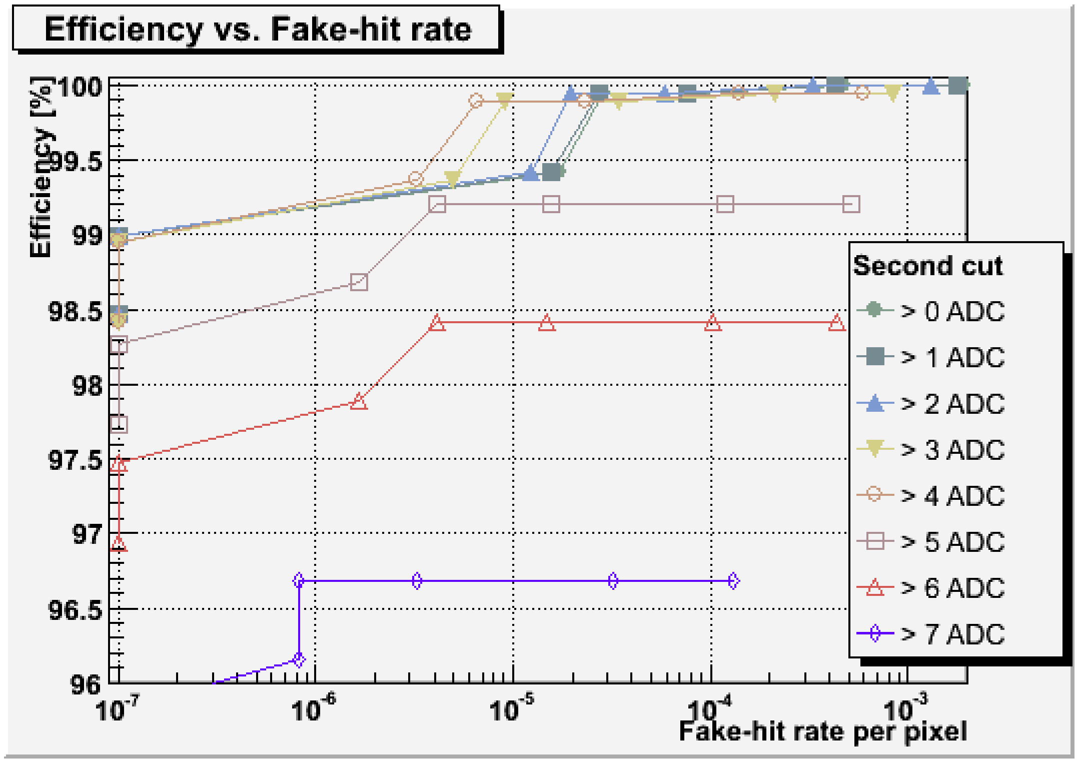

Figure 5 Efficiency vs. fake-hit rate per pixel for the cluster finding algorithm based on two thresholds: high cut on the central pixel and lower cut on one of 8 adjacent pixels.

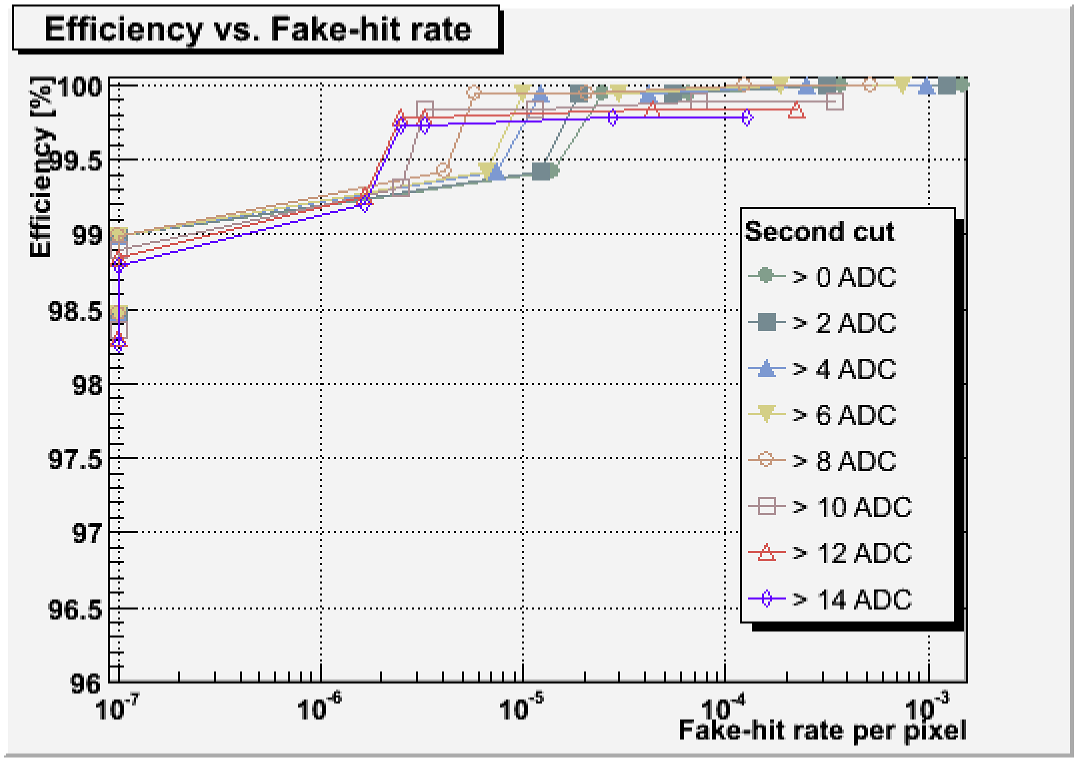

Figure 6 Efficiency vs. fake-hit rate per pixel for the cluster finding algorithm based on two thresholds: high cut on the central pixel and the second cut on sum of signals from all of 8 adjacent pixels.

For the satisfactory performance of a detector i.e. a high detection efficiency (>99 %) and a low fake-hit rate per pixel ($<10^{-4}$) are required. It can be seen that the proposed two-threshold algorithm meets these criteria for a various pairs of the two cuts i.e. center pixel threshold ≥10 ADC and low threshold < 5 ADC. This should allow settings to be chosen that comfortably reach the desired performance.

Although the algorithm based on sums of signals is much more flexible and the range of acceptable settings is much wider, the proposed two-threshold algorithm is more suitable to the HFT readout system. This is due to a very good performance achievable with an extremely simple architecture. This simple architecture can be very easily implemented in an FPGA and also seems to be an attractive solution for on-chip implementation.

Additional considerations:
In this study, a variety of cluster finders based on two different thresholds was investigated. Considered algorithms had the same cut on the central pixel but different cuts on the neighbor pixels:

- Any one pixel of 8 neighbors passes the lower threshold (described above),
- Any two pixels of 8 neighbors pass the lower threshold,
- Any three pixels of 8 neighbors pass the lower threshold,
- Any 2 adjacent pixels of 8 neighbors pass the lower threshold,
- Any 1 pixel of 4 neighbors passes the lower threshold (Figure 7b),

- Any 2 pixels of 4 neighbors pass the lower threshold (Figure 7b).

Compared to the first algorithm, all of them are much more rigorous. For higher subset of lower threshold values they significantly reduce fake-hit rate, but at the same time the efficiency is significantly affected and drops below 99%. This allows only a more limited combination of settings that would meet the stringent requirements.

Probably the most interesting algorithm of those mentioned above is the one that checks the lower threshold against only 4 pixels adjacent to the central pixel, as presented in Figure 7b. The efficiency and fake-hit rate are presented in Figure 8. If it is compared with the originally proposed algorithm it shows a fake-hit rate decrease by a factor of 2 with only a small penalty in efficiency ~0.1%.

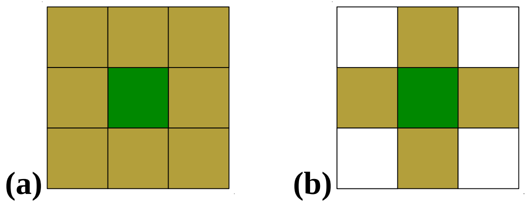

Figure 7 Topology of cluster finding algorithms that check eight (a) and four (b) adjacent pixels.

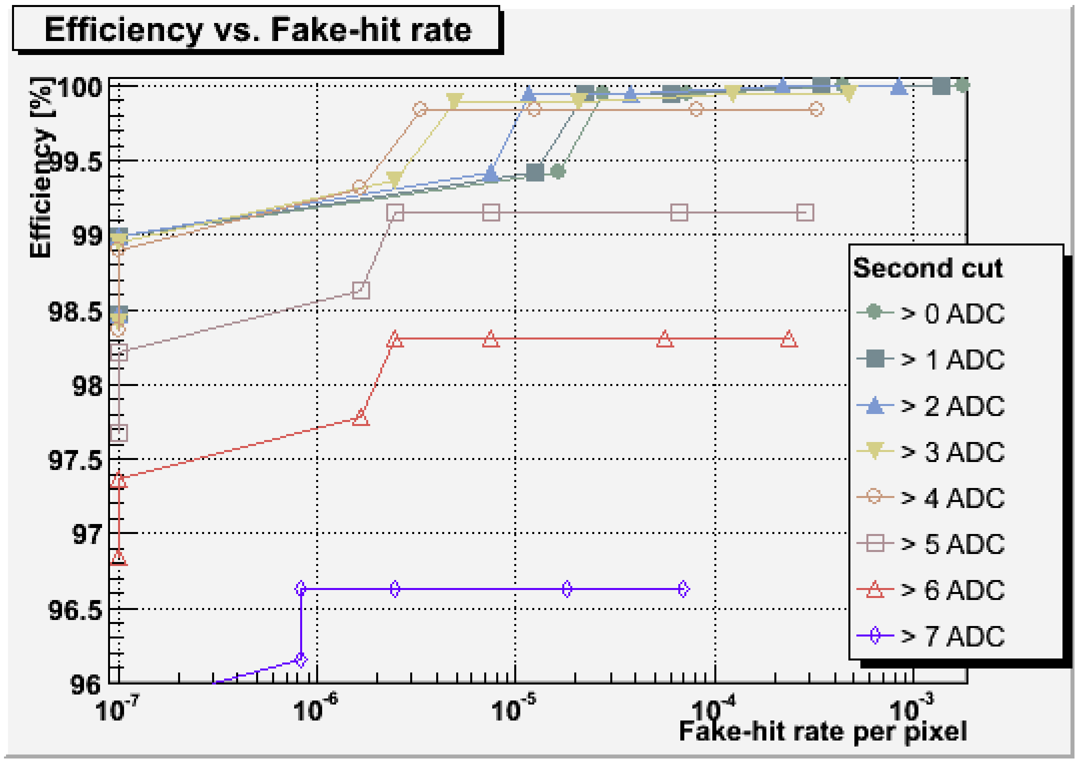

Figure 8 Efficiency vs. fake-hit rate per pixel for the cluster finding algorithm based on two thresholds: high cut on the central pixel and lower cut on one of 4 adjacent pixels forming a cross pattern.