Thus the $\lesssim 3\%$ interesting data should be selected at the hardware level while the rest, likely generated by the noise, should be suppressed. These operations are performed by the *FEROMs*, i.e. the Front End Read Out Modules, that receive the analogue data from the detector electronics [50]. The main FEROM task is the digitization of the 2.6 million SSD strip values in 160 μs. This is achieved by digitizing in parallel the 1698 double-sided SSD module signals (each containing 1536 strip values). The FEROM also performs the offset correction (pedestal) and the zero-suppression, the event-data sorting using Multiple Event Buffers and the event data transfer to the DAQ.

The FEROM system consists of 8 crates, each containing 216 analogue-to-digital converters and the interfaces connected to the data acquisition system, the detector control system and the central trigger processor. The scheme in fig. 4.1 describes the complete acquisition chain, with the FEROM modules connecting the front-end electronics with the data acquisition system. The digitized data are corrected for pedestals and zero-suppressed for each event individually; the remaining data are stored in a buffer and sent to the data acquisition system. In order to properly correct the data, i.e. achieving the occupancy reduction without losing interesting physical signals, the FEROM applies an additional algorithm to suppress the base-line distortions. These distortions are mainly generated by the so-called *common mode noise* that is discussed in the next paragraph. The FPGA based design allows easy upgrading of the algorithm for the correction of this particular effect.

4.2 The common mode noise.

Besides the noise sources intrinsically related to the sensor and to the front-end electronics, the oscillation at a channel input can be enhanced by environmental origins, like by the pick-up effect of the sensor and by the instability of the reference grounds induced by the power supplies; this component of the oscillation is called *common mode* shift. While the intrinsic noise is characteristic of the single channel and the event-by-event oscillation of its output does not depend on the other channels, the common mode noise influences coherently a group of neighboring channels. Due to this fact, it's possible to estimate the common mode component in each event and correct it.

Normally, it is *common* to the smallest group of channels read by the same front-end chip, even if sometimes some substructures can appear in its distribution. In the ALICE SSD case, the common mode noise causes a coherent mostly Gaussian distributed oscillation of the baseline of the pedestals of 128 channels read by the same HAL25 chip. In the following paragraph the sources of this noise component are discussed.

4.2.1 Sources of common mode noise.

The common mode noise can be originated both from internal and external electromagnetic sources. It derives from ground loops in the power supplies used for the detector and the ADC; another contribution comes from the silicon strips and the cable lines acting like antennas in a radio frequency field. As experienced analyzing the noise data, the common mode noise is the sum of different oscillations with different amplitudes and frequencies.

An extensive optimization work was led in order to minimize the contribution introduced by the power supplies specifically developed by CAEN for the ALICE SSD. The optimization of the electronic components and of the electrical characteristics allowed to reduce the common mode σ down to 5 ADC rms units.

In the experimental site, the environmental conditions make the common mode noise grow up to a typical value of 7 ADC rms units, as it has been measured during a large number of noise runs involving all the sub-detectors. This is probably due to the presence of many active electronic devices close to the SSD modules and to the induced antenna-effect.

In fig. 4.2, the typical noise as measured on an SSD module installed in the experimental site: the plots show the total noise (*left panel*), the intrinsic noise calculated after the common mode subtraction (*central panel*), and the common mode noise for all the channels of the module (*right panel*) as a function of the strip number.

4.3 The Common Mode correction.

4.3.1 The common mode shift calculation.

In high energy physics experiments, various methods have been applied to correct the common mode noise contribution to the signals read by a group of neighboring

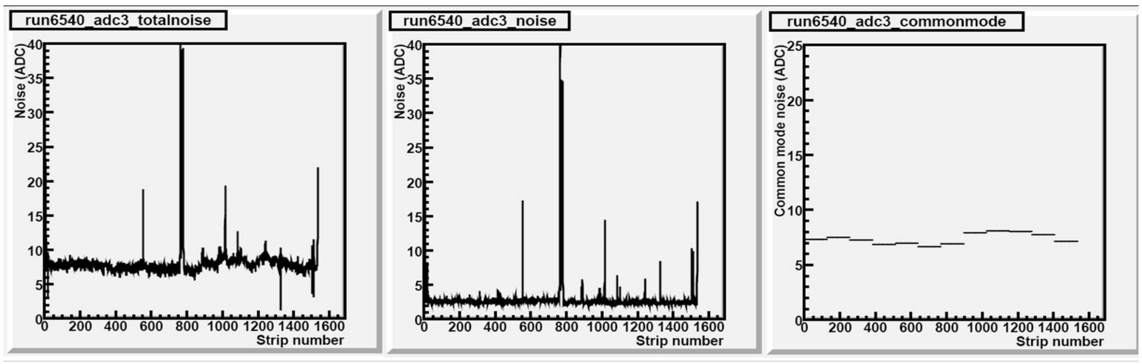


Figure 4.2: The noise affecting an SSD module in the final location. The various component of the noise measured on the 1536 channels of an SSD module are showed in the plots: the total noise (left), the common mode corrected noise (center) and the common mode noise (right). All the channels belonging to the same chip are characterized by the same common mode noise; it oscillates from chip to chip around the value of 7 rms units.

channels of a detector. Since it is a coherent displacement of the pedestals of all the channels belonging to the same chip, this baseline shift can be calculated averaging the signal amplitude of the 128 channels of a chip. Hereafter the simplest definitions of the pedestal, the noise and the common mode shift are briefly summarized.

Pedestal, noise and common mode determination.

The *pedestal* represents the read-out value of a channel in absence of both particle signals and noise. Normally it doesn't correspond to the origin of the ADC scale and can vary from channel to channel. It is mainly related to the presence of a DC offset at the output of the read-out chips and it has to be subtracted to properly evaluate the signal amplitude. In order to compute the pedestal and the noise for the i-th channel, a sample of M read-out data without particle signals at the input is needed. The pedestal is a simple mean of the digitized data:

$$P_i = \frac{1}{M}\sum_{j=1}^{M} ADC_{ij} \tag{4.1}$$

where ADC_{ij} represents the j-th read-out value of the i-th channels. The read-out data have gaussian distribution with standard deviation:

$$R_i^{tot} = \sqrt{\frac{1}{M-1}\sum_{j=1}^{M}(ADC_{ij} - P_i)^2} \tag{4.2}$$

that is the i-th channels total noise.

The simplest common mode calculation algorithm is described by the equation 4.3

$$CM_{(l)j} = \frac{1}{N_{spy(l)}} \sum_{i_{spy(l)}=1}^{N_{spy(l)}} (ADC_{i_{spy(l)}j} - P_{i_{spy(l)}}) \tag{4.3}$$

where:

- $CM_{(l)j}$ is the *common mode* of the l-th chip in the j-th event
- $N_{spy(l)}$ is the total number of spy channel (i.e. the channels considered in the calculation) of the l-th chip
- $ADC_{i_{spy(l)}j}$ and $P_{i_{spy(l)}}$ are respectively the j-th read-out value and the pedestal of the i-th spy channel belonging to the l-th chip.

This basic algorithm is very efficient when applied to data in absence of particle signals and to channels showing a homogeneous gaussian intrinsic noise σ.

Channels presenting high pulses due to an abnormal noise fluctuation or to a particle detection, can indeed introduce a significant error in the comon mode calculation. Thus, the simple average is not safe when applied to physics event acquisition runs.

4.3.2 Benchmark algorithm for the CM correction

In order to test the performances of the common mode corrections algorithms presented in this chapter, a benchmark algorithm has been designed: all the common mode calculations presented in the this chapter are compared with the results obtained with this *BestCM* algorithm.

Supposing not to have time and computing resources limitations, contrary to the real case, this algorithm analyzes the same data more than once, applies selections and calculate averages and standard deviations. Due to the complexity of the operations performed, these calculation are in fact not allowed in the acquisition hardware, but give a result that aims at being considered a good approximation of the real common mode shift.

The *BestCM* algorithm acts in two steps over the signals collected in an event by the 128 channels of a chip. The channels corresponding to the first and to the last 16 strips of the SSD module are excluded from the calculations, because of their different behaviour in terms of common mode noise, as will be explained in the

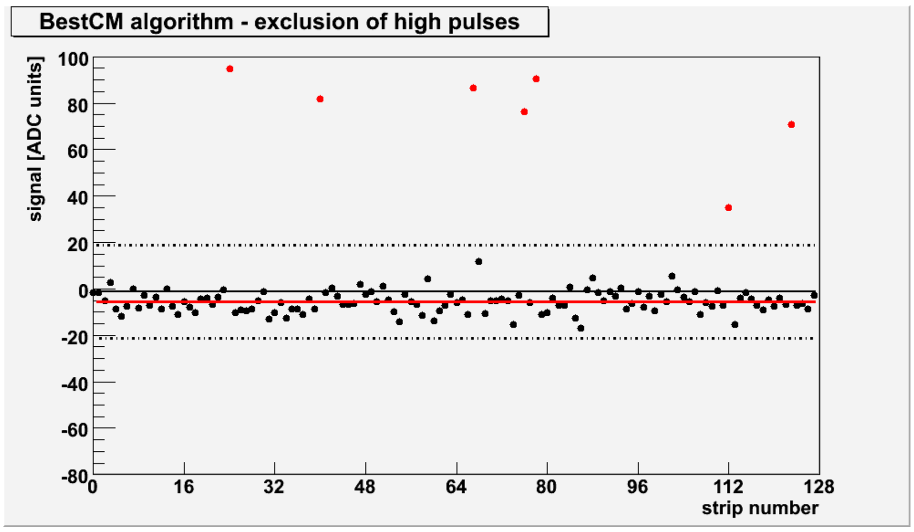


Figure 4.3: The plot reports the signals measured by the 128 channels of a chip. A first average of all the 128 read values gives the mean shown by the black solid line. The RMS of their distribution is used to define a range centered on this mean and extending between the dotted lines. The points falling outside this range (red points) are excluded and the mean of the remaining signals is successively calculated (red line). The resulting value defines the *BestCM* common mode shift.

following paragraph. In the first step the algorithm calculates the average and the σ of the signals distribution. The channels presenting a signal far from the average by more than 2σ are rejected and not used in any further calculation.

In this way, in the second step, strips hit by a particle or channels presenting a large oscillation due to noise are excluded. The unrejected signals are averaged again: the result is the *'true'* common mode correction (*BestCM*). The procedure is visualized by the scheme in fig. 4.3 for a central chip of a module.

These operations guarantee the independence of the result from unexpected single channels noise oscillations and from particle signals, even in case of high occupancy of the chip.

4.3.3 CM correction for the first and last strips.

Analyzing in detail the common mode oscillations, a particular behaviour has been noticed on the first and the last strips of each module side. These strips differ from the central strips for geometrical and electric characteristics: due to their position in the sensor, they are coupled to a capacitive network which is different with respect to the central strips; moreover, they partially face a region of the sensor backplane that is not connected to the front-end electronics and that therefore presents different properties; some of these strips are also shorter because of their inclination with respect to the module edge. The sum of these factors induces on the corresponding channels a different response to the common mode noise.

The correlation between the single channel readout with the common mode shift decreases moving toward the edges of the module, as clearly visible in fig. 4.4: for each of the first 12 strips of a module P-side, the signal read in 500 events is plotted as a function of the calculated common mode (*BestCM*) in such events. Therefore these channels should not be taken into account by a proper common mode calculation with the same weight of the central strips. The possibility to correct

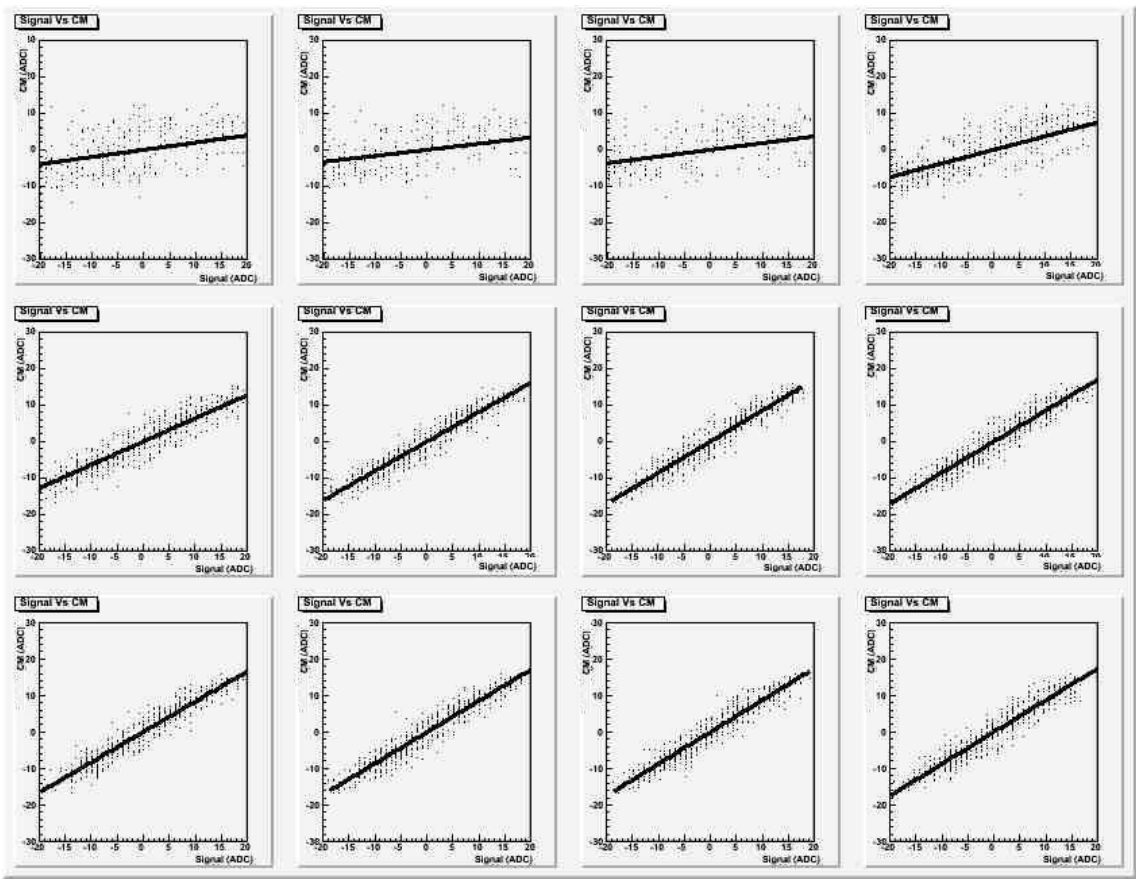


Figure 4.4: The correlation between the common mode shift and the signal measured by the single channel, plotted for the first 12 channels of an SSD module. The correlation increases while moving toward the central strips.

properly the common mode shift in the first and last channels was explored. Since the common mode correction is aimed at the reduction of the read-out oscillations, i.e. of the noise, the dependence between the common mode subtracted noise and the correction applied in each event was studied for each channel.

The most efficient method to estimate the right correction turned out to be the minimization of the noise as a function of the readout signal, the *BestCM* and a scale factor between the common mode and the read-out signals of each channel. Supposing the common mode to have normal distribution with $\mu = 0$, the noise after the common mode correction for the i-th channel can be expressed by the equation 4.4:

$$R_i = \sqrt{\sum_{j=1}^{M} \frac{(ADC_{ij} - \alpha_i \cdot CM_{(l)j} - P_i)^2}{M-1}} \tag{4.4}$$

where

- R_i is the common mode corrected noise for the i-th channel
- ADC_{ij} is the j-th value read-out by the i-th channel
- $CM_{(l)j}$ is the *common mode* of the l-th chip in the j-th event
- P_i is the i-th channel pedestal
- α_i is the scale factor.

The common mode value has to be multiplied for this α coefficient, which is plotted as a function of the strip number of a module in fig. 4.5, in order to be properly corrected even for the lateral strips. It is $\sim$ 1 for the central strips and for the properly working channels, the ones used for the common mode calculation, while it grows up to $\sim$ 10 in the lateral strips. The plot in fig. 4.6 shows the value of the coefficient for the strips lying close to the module edges (last P-side channels and first N-side channels).

Correcting with these factors the common mode in each event and for each strip, its subtraction can be improved in order to obtain a better estimate of the intrinsic noise: as experienced by applying this method to a sample of modules, for the strips placed on the edges it is possible to obtain a reduction of about 5-10 rms ADC-units for the noise after the subtraction, if compared with the unweighted correction, bringing them back to an acceptable noise level. The stability of the scale factors, which have been calculated on a large event statistics, was tested with good results on different data samples: therefore they can be calculated just once and then implemented in the FPGA in order to perform the improved correction at the hardware level.

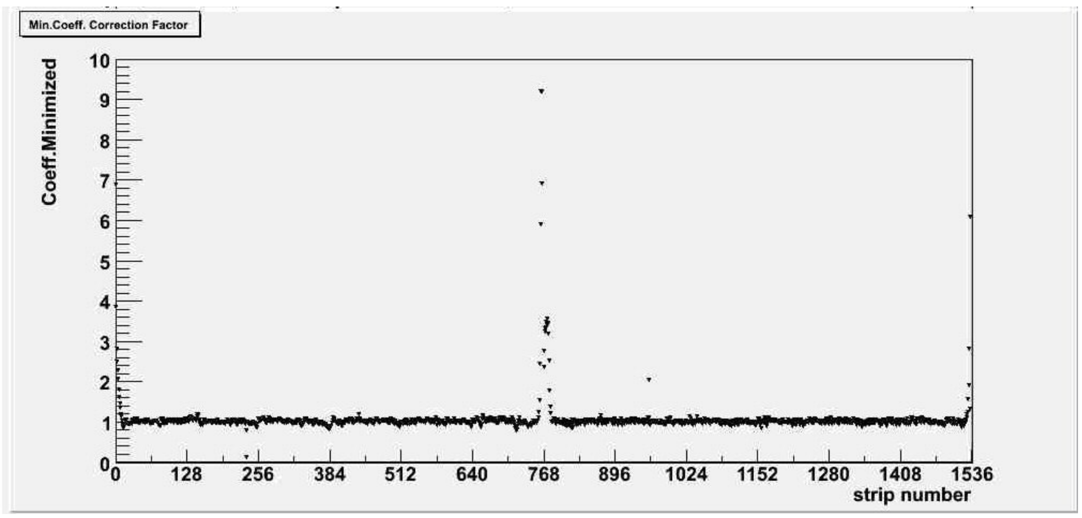


Figure 4.5: The common mode correction factor: the scale factor is plotted as a function of the 1536 channels of a module; multiplying this factor, that has been calculated for this specific module using the equation 4.4, by the common mode shift value, it's possible to correct better the signals and minimizes the intrinsic noise as regards these channels. Note that the 767 is the last channel of a module P-side, while 769 is the first of the N-side.

4.4 The current FEROM algorithm.

The algorithm implemented in the acquisition hardware during the commissioning phase, called *FastCM* algorithm, differs from the simple average over the chip channels belonging to the same chip only for a particular: it excludes the first and last 16 channels of each chip and applies a mask based on a static map in order to reject the channels classified as *bad*. This map contains all the channels presenting high noise, namely $\sigma > 20$ *ADC units* (*noisy*), whose front-end electronics is inactive (*dead*) or whose connection to the read-out is broken (*open*), as results from the analysis previously performed to calibrate the detector. Only 64 *good* channels belonging to the central region are used for the common mode calculation.

This algorithm filters the signals taking into account the well known defects, but it is unsafe for what regards unexpected high-noise events and particle events, whose position is unpredictable.

In order to test its performances, the residuals with respect to the 'true' common mode shift have been calculated for 500 real noise events and used to fill an histogram. The resulting distribution measures the goodness of the algorithm in the specific case. The performance test has been carried out on a large number of modules belonging to different ladders.

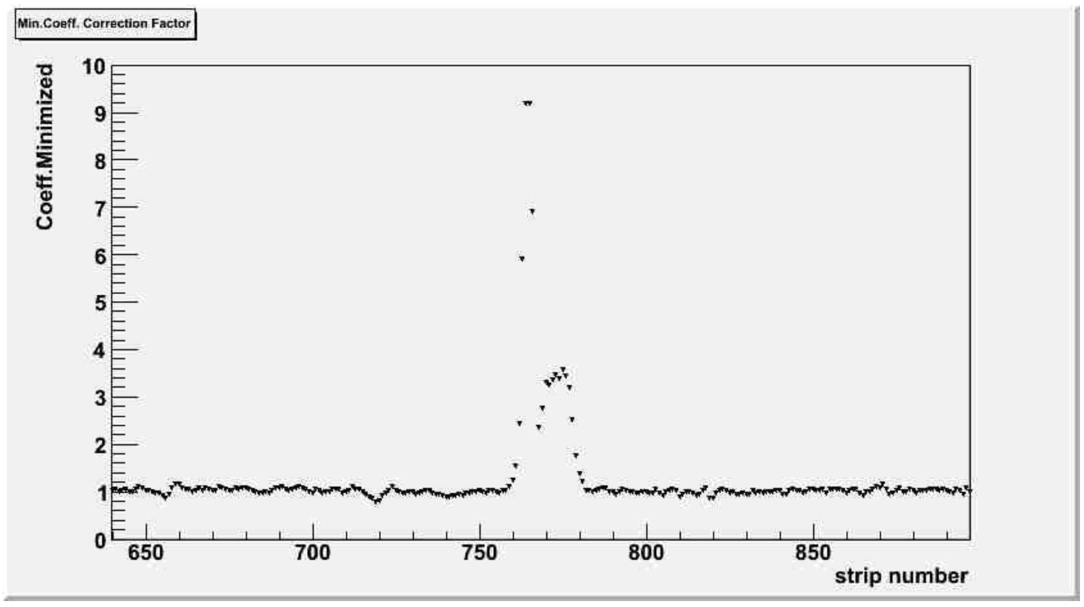


Figure 4.6: The scale factor for the common mode correction for the channels lying near the module border. It reaches the value of 10, while for the channels corresponding to the central region of the sensor is about 1.

4.4.1 The algorithm performances.

For chips presenting a homogeneous noise σ not larger than 7 rms units, the algorithm shows very good results: the differences are distributed around zero with a $\sigma < 1$ (left histogram in fig. 4.8).

In the case of a chip hosting some channels with a very large noise (up to 100 rms units in the histogram shown in the left panel of fig. 4.7), the simple average would miscalculate the common mode shift by a quantity that depends on the signals from these noisy channels while the implemented algorithm calculates properly the shift. The large distribution of the errors made with a simple average is shown in the left histogram. On the right the results of the actual algorithm: the mask applied to the bad channels is very efficient and the errors are negligible (smaller than 2 ADC units in the 98% events).

Finally, the presence of particle signals has been simulated adding on the same 500 noise real events some pulses randomly distributed over the 64 central channels. The simulation of 5 MIPs equivalent signals (5×140 ADC units) produces the effect shown in the right histogram in fig. 4.8, i.e. a mean residual of about 11 ADC units. Therefore, the presence of particle signals introduces a systematic error, related to the number of particles detected by the strips considered for the *FastCM* calculation.

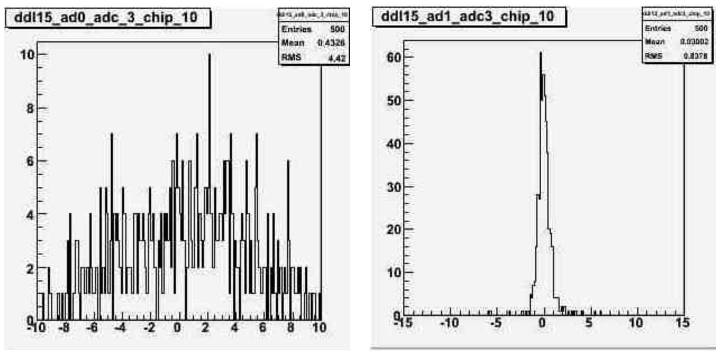


Figure 4.7: Residuals between the *true* common mode and the calculated one in presence of several noisy channels (with σ up to 100 ADC rms units) with *left panel* or without *right panel* the use of a mask to reject the *bad* channels.

The result is mostly independent of the clusters shape.

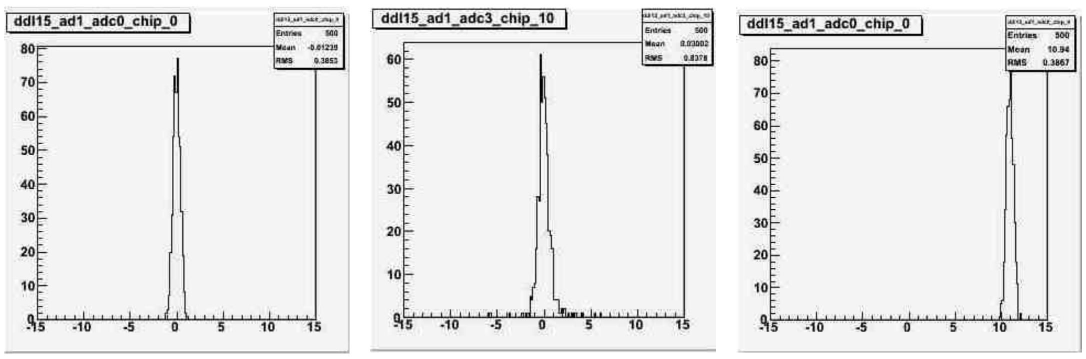


Figure 4.8: Performances of the common mode algorithm currently implemented in the SSD acquisition system: residuals with respect to the *true* common mode for: a chip with typical noise (uniform $\sigma \sim 4\, adc\ units$) and without particle signals *(a)*, very noisy chip (some channels with $\sigma > 15\, adc\ units$) without particle signals *(b)*; a chip in presence of particle signals *(c)* corresponding to 5 MIP. In the last case a systematic displacement can be noticed.

4.5 Proposals of two algorithms for the common mode correction.

In order to improve the algorithm efficiency in presence of particle signals, some other algorithms have been considered. They were evaluated in terms of performances, time consumption and simplicity of the executed operations.

The *multi step* algorithm, used in this work as benchmark algorithm in the *two steps* version, needs the data to be processed more than once to select the proper channels for the common mode calculation. This request can not be granted due to the architecture of the FPGA firmware that allows to process the data only once.

The *fast ascending* method, used e.g. for the readout chain of the CMS silicon Preshower detector, is based on simple operations and assured good results under different conditions (common mode shift calculated for groups of 16 channels). It can perform an on-line sorting of channels by pulse height and reject signals from particles [51]. This algorithm was considered not compliant with the hardware constraints by the FEROM firmware developers because of the large amount of channels to treat in the SSD case.

Eventually, two algorithms have been considered: the first is quite simple and consists in the introduction of a fixed threshold to exclude those channels hit by particles or simply presenting a high pulse. A possible more complex evolution of this method allows to solve most of the drawbacks of the *fixed threshold* algorithm: it is based on a *self tuning* procedure.

4.5.1 The *fixed threshold* algorithm.

A simple improvement of the *FastCM* algorithm implemented in the data acquisition system during the detector commissioning phase consists in the introduction of a fixed threshol, which rejects the high pulses from the channels not yet masked. The value chosen as threshold takes into account the typical common mode noise distribution in the SSD and minimizing the effects of the particle signals.

- **Algorithm description.** It considers only the central channels (from the 17th to the 112th strip) excluding the bad channels; among the unrejected, it accepts the signals below the threshold and averages them.

- **Performances.** The histogram in fig. 4.9 shows the distribution of the errors made by the algorithm, when a fixed 40 ADC units threshold is applied; i.e. rejecting all signals whose values are larger then 40 ADC units; the results are encouraging for all considered cases: in absence of particle signals, it reproduces the performances of the *FastCM* algorithm: the error is negligible both with and without the presence of noisy channels.

In presence of particle signals, the 40 ADC units threshold cuts most of the simulated particle signals: in case of a MIP, it is completely rejected when collected by one single strip, while in case of a multistrip cluster, only the tails can survive, if smaller than 40 ADC units. Also in case of high local particle multiplicity, i.e. with one particle per cm^2, the simulations show a residual smaller than 3 ADC units.

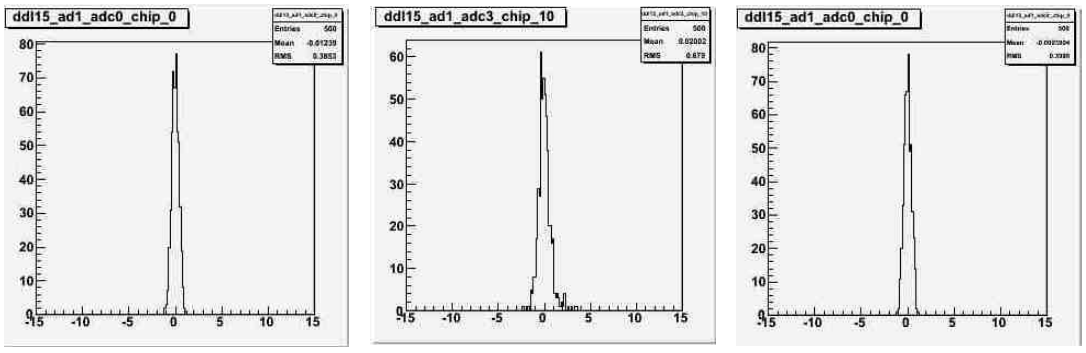


Figure 4.9: Peformances of the *fixed threshold* algorithm. The histograms show the residuals with respect to the *true* common mode for: a chip with typical noise (uniform $\sigma \sim 4$ *adc units*) and without particle signals *(a)*, very noisy chip (some channels with $\sigma > 15$ *adc units*) without particle signals *(b)*; a chip in presence of particle signals *(c)* corresponding to 5 MIP. In all these cases, the residuals are tightly distributed around zero.

- **Advantages and drawbacks.** The simplicity of the algorithm is preserved, since the comparison with the threshold is the only additional operation introduced by this algorithm. On the other side, the *fixed threshold* algorithm has some drawbacks: the threshold is not very tight, so that at least the tails of the multistrip clusters can influence the calculation; moreover, this algorithm rejects completely the events with a common mode larger than 40 ADC units. During noise tests, such a large shift of the baseline was rarely observed, since the typical common mode σ is equal to 7 rms units for the SSD modules.

  Moreover, the algorithm efficiency in discarding the particle signals strongly depends on the common mode magnitude and sign: while the threshold is fixed, the common mode oscillates so that the threshold effectiveness changes event by event.

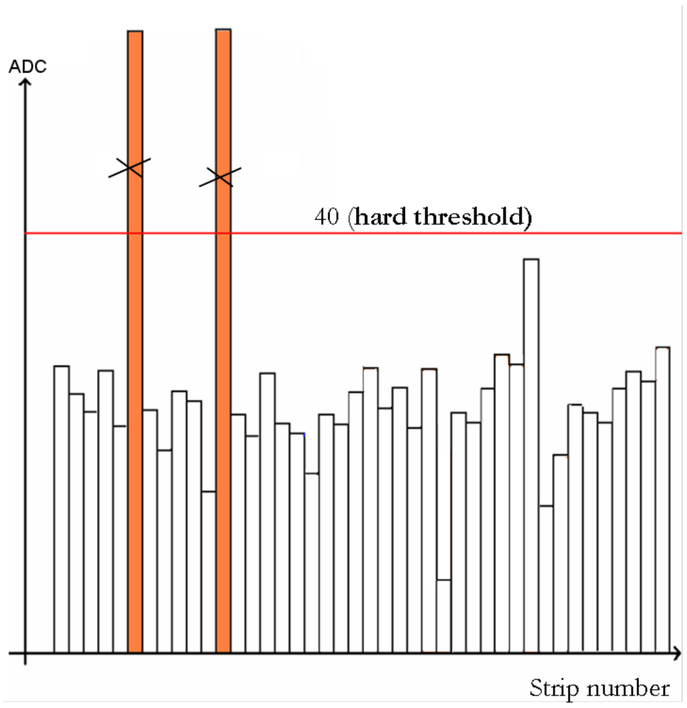


Figure 4.10: The *fixed threshold* algorithm: the signals above the 40 ADC units threshold (*red line*) are rejected and the common mode shift is calculated as the average of the remaining channels.

4.5.2 The *self tuning* algorithm.

In order to reduce further the errors and to avoid the loss of data in case of events with a very large common mode displacement of the channel baseline, another algorithm was designed and tested. It is structured into two phases, even if it processes the data just once. The *self tuning* procedure is described in the scheme reported in fig. 4.11.

- **Algorithm description.** The first and last 16 channels of the chip are excluded from the calculation. In the first part it calculates the common mode as a simple average (m) of the signals coming from 16 out of 32 strips (from the 17th to the 48th strips) and smaller than a threshold $T = 50\,ADC\ units$; this threshold has a quite large value, in order to exclude only the biggest signals and to include data with a large CM, reducing the risk to completely lose an event. In the second part it considers only the strips from the 49th to the 112th and takes into account only the first 32 signals whose amplitude S falls within a tight range around the previously calculated m, namely if it satisfies the condition $m - r < S < m + r$, with $r = 10\,ADC\ units$; the common mode

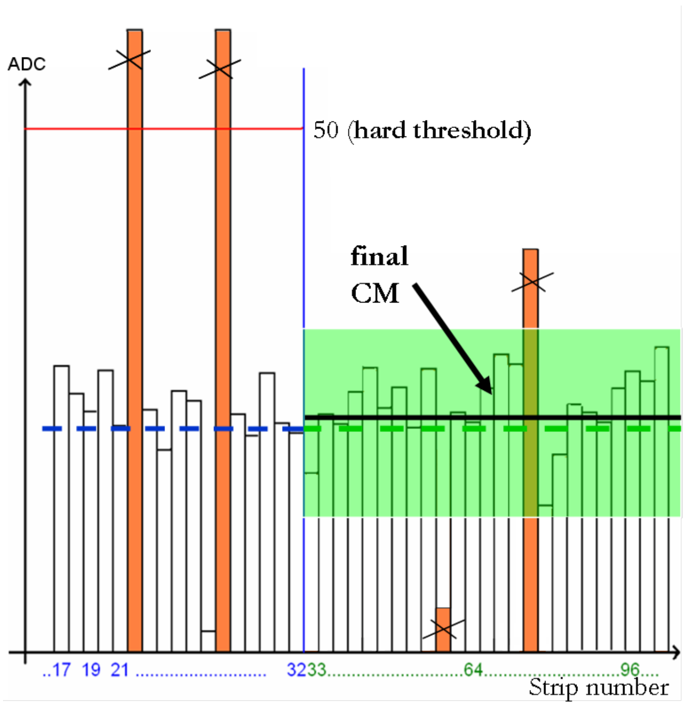


Figure 4.11: The *self tuning* algorithm operations: *a)* a fixed 50 units threshold (*red line*) applied on a first set of channels rejects very high signals; *b)* the other signals are averaged (*dashed blue line*); *c)* among the second channel set, only the signals within a certain tight window around the previously calculated mean are accepted (*green range*); *d)* finally, the average of the accepted signals determines the common mode shift of the event (*black solid line*).

shift is calculated as the average of the accepted signals.

- **Parameters tuning.** A study was carried out in order to optimize the parameters used by this algorithm:

  - The particles signals are simulated on the hypothesis of high particle multiplicity foreseen in heavy ion collisions: a further optimization will be possible after first measurements of this multiplicity on the basis of real data.

  - The number of strips included in the first and in the second group depends on the multiplicity of hits and of noisy strips on that zone of the detector and it has been tuned in order to minimize the error in the common mode evaluation and to maximize the number of strips used in every calculation. With the considered multiplicity, we obtained always at least 16 good strips in the first step and 32 in the second step, so that we always managed to calculate the common mode over a large sample.

  - As already explained, the 50 ADC units threshold was chosen to accept events with large baseline displacements.
  - The range defined by the parameter $r = 10$ *units* has been tuned according to the typical dispersion of the signal amplitudes in absence of physical events.

- **Performances**: the *self tuning* algorithm gives good results in presence of noisy channels and physical signals, accepting common mode shifts as high as 50 units; it should be noted, anyway, that such a large common mode shift appears to be rare in the SSD environment and it mostly affects chips presenting also an uncommon intrinsic noise behaviour; therefore it can be hardly corrected. As shown in the left and central histograms of fig. 4.12, it gives very good results without particle signals. It is also very robust in presence of particle signals, even in case of very high multiplicity: even in presence of multistrip clusters, the tails accepted by the window can introduce an error smaller than 1 ADC unit. The *self tuning* algorithm is compatible with the mask for bad channels. But can give good results also without that mask. Contrary to the *fixed threshold* case, the present algorithm effectiveness is rather stable and independent of the common mode magnitude. Moreover, it preserves the characteristic of simplicity, since it adds only an average calculation and comparison operations to the algorithm.

4.5.3 Common mode correlation between neighboring chips.

The case of a failure of the common mode correction performed at the hardware level is here considered.

The algorithm class discussed before bases the common mode calculation on the channels selected through a system of thresholds: in some particular cases, such an algorithm can fail to reach the minimum required number of channels to calculate the oscillation. This type of failure can happen for example:

- in the events with a very large displacement of the baseline, which turns out to bring the signals above the threshold;

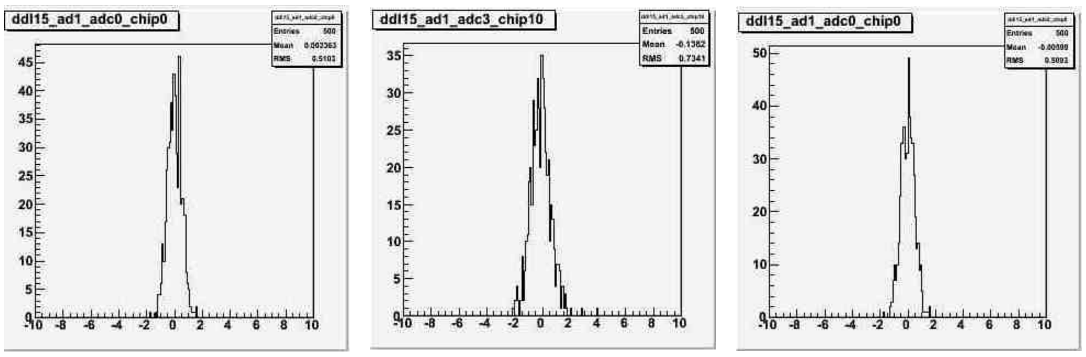


Figure 4.12: Peformances of the *self tuning* algorithm. The histograms show the residuals with respect to the *true* common mode for a chip with typical noise (uniform $\sigma \sim 4$ *adc units*) and without particle signals *(a)*, very noisy chip (some channels with $\sigma > 15$ *adc units*) without particle signals *b*; a chip in presence of particle signals *(c)*. In all these cases, the residuals are distributed around zero with a $\sigma < 1$.

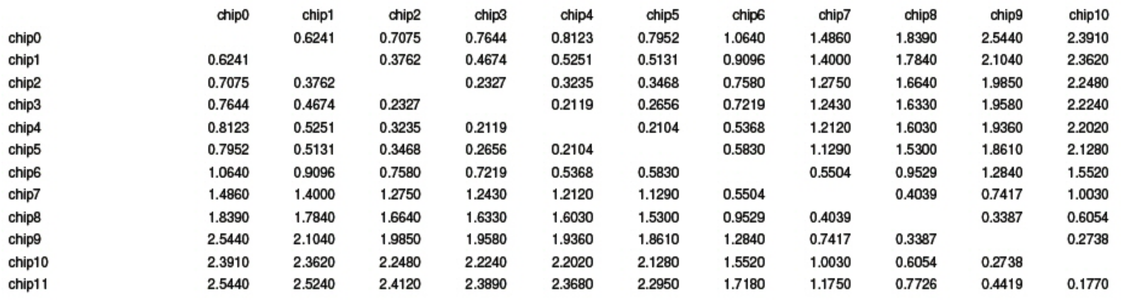
- for chips presenting a high local particle hit multiplicity, where many channels are then rejected;
- in drastically disturbed events.

In these cases, it is fundamental to switch off the usual common mode correction and to try to recover an estimate of the common oscillation basing the calculation on a larger group of channels: with this purpose, the common mode oscillations of a chip has been studied as a function of the oscillation of the other chips belonging to the same module.

In order to take into account the possible influence of the constructive characteristics of the SSD sensor on the common mode oscillations, the behaviour of modules of all the three different manufacturers has been observed. All the considered samples show a linear dependence between the oscillations of different chips, with a linear coefficient close to ~ 1 for first-neighboring chips. Therefore a calculation failure in a chip can be solved extrapolating the common mode value from a neighboring chip.

In case of loss of informations about neighboring chips, it is even possible to take the common mode corresponding to a chip from the opposite module-side, with a maximum statistical σ of 2 ADC rms units. In the table (fig. 4.13) we can read the RMS of the distribution of the difference between common mode oscillations of different chips, over a statistics of 500 events for a module. We found a similar behaviour in all the modules we analyzed, including all manufacturer-types and the noisy modules. Moreover, the RMS of the difference between j-th and *(j+1)*-

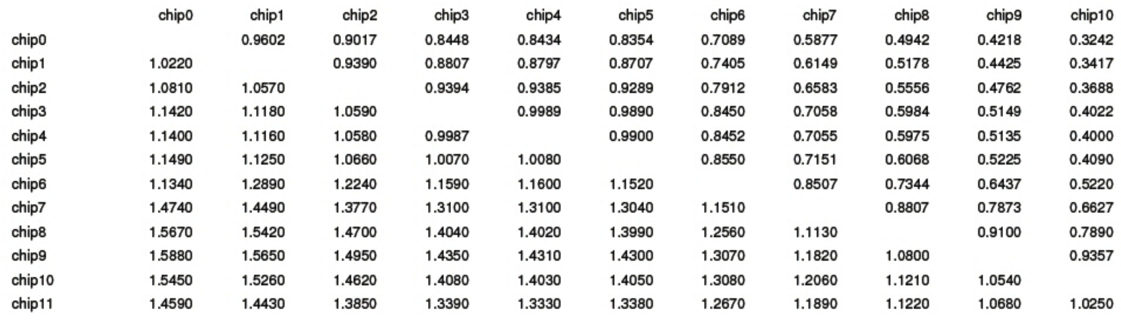
th chips is always smaller than 1 unit (see along the diagonal). The coefficient of



	chip0	chip1	chip2	chip3	chip4	chip5	chip6	chip7	chip8	chip9	chip10
chip0		0.6241	0.7075	0.7644	0.8123	0.7952	1.0640	1.4860	1.8390	2.5440	2.3910
chip1	0.6241		0.3762	0.4674	0.5251	0.5131	0.9096	1.4000	1.7840	2.1040	2.3620
chip2	0.7075	0.3762		0.2327	0.3235	0.3468	0.7580	1.2750	1.6640	1.9850	2.2480
chip3	0.7644	0.4674	0.2327		0.2119	0.2656	0.7219	1.2430	1.6330	1.9580	2.2240
chip4	0.8123	0.5251	0.3235	0.2119		0.2104	0.5368	1.2120	1.6030	1.9360	2.2020
chip5	0.7952	0.5131	0.3468	0.2656	0.2104		0.5830	1.1290	1.5300	1.8610	2.1280
chip6	1.0640	0.9096	0.7580	0.7219	0.5368	0.5830		0.5504	0.9529	1.2840	1.5520
chip7	1.4860	1.4000	1.2750	1.2430	1.2120	1.1290	0.5504		0.4039	0.7417	1.0030
chip8	1.8390	1.7840	1.6640	1.6330	1.6030	1.5300	0.9529	0.4039		0.3387	0.6054
chip9	2.5440	2.1040	1.9850	1.9580	1.9360	1.8610	1.2840	0.7417	0.3387		0.2738
chip10	2.3910	2.3620	2.2480	2.2240	2.2020	2.1280	1.5520	1.0030	0.6054	0.2738	
chip11	2.5440	2.5240	2.4120	2.3890	2.3680	2.2950	1.7180	1.1750	0.7726	0.4419	0.1770

Figure 4.13: The RMS of the distribution of the difference between the common mode oscillations of different chips belonging to the same module (500 events sample).

linear dependence between the common mode of different chips can be considered to calculate the possible error also in case of larger noise: taking the neighboring chip common mode it can generate a maximum relative error of about 15% (see fig. 4.14).



	chip0	chip1	chip2	chip3	chip4	chip5	chip6	chip7	chip8	chip9	chip10
chip0		0.9602	0.9017	0.8448	0.8434	0.8354	0.7089	0.5877	0.4942	0.4218	0.3242
chip1	1.0220		0.9390	0.8807	0.8797	0.8707	0.7405	0.6149	0.5178	0.4425	0.3417
chip2	1.0810	1.0570		0.9394	0.9385	0.9289	0.7912	0.6583	0.5556	0.4762	0.3688
chip3	1.1420	1.1180	1.0590		0.9989	0.9890	0.8450	0.7058	0.5984	0.5149	0.4022
chip4	1.1400	1.1160	1.0580	0.9987		0.9900	0.8452	0.7055	0.5975	0.5135	0.4000
chip5	1.1490	1.1250	1.0660	1.0070	1.0080		0.8550	0.7151	0.6068	0.5225	0.4090
chip6	1.1340	1.2890	1.2240	1.1590	1.1600	1.1520		0.8507	0.7344	0.6437	0.5220
chip7	1.4740	1.4490	1.3770	1.3100	1.3100	1.3040	1.1510		0.8807	0.7873	0.6627
chip8	1.5670	1.5420	1.4700	1.4040	1.4020	1.3990	1.2560	1.1130		0.9100	0.7890
chip9	1.5880	1.5650	1.4950	1.4350	1.4310	1.4300	1.3070	1.1820	1.0800		0.9357
chip10	1.5450	1.5260	1.4620	1.4080	1.4030	1.4050	1.3080	1.2060	1.1210	1.0540	
chip11	1.4590	1.4430	1.3850	1.3390	1.3330	1.3380	1.2670	1.1890	1.1220	1.0680	1.0250

Figure 4.14: The coefficients of linear dependence between the common mode values of different chips belonging to the same module.