Draft
CMS/99-000
October 8, 1999

Reduction of ECAL data volume using lossless data compression techniques

Ph. Busson, A. Karar, G.B. Kim
L.P.N.H.E.-Ecole Polytechnique, IN2P3-CNRS, Palaiseau, France
and
D.W. Kim, S.C. Lee
Department of Physics, Kangnung National University, Kangnung 210-702, South Korea

Abstract

We investigate the possibility of reducing the data size of the electromagnetic calorimeter(ECAL) of CMS. The Selective Readout is applied first to reduce the data size at a manageable level. Then various data compression methods are considered, and their performances are estimated using the data from the full simulation of the ECAL system. A reduction of the average event size by a factor of two or larger is obtained in most of the cases.

1 Introduction

The electromagnetic calorimeter(ECAL) of the CMS detector consists of 82,728 lead tungstate ($PbWO_4$) crystals. Each crystal has a square cross section of 22×22 mm^2 covering $\Delta\eta \times \Delta\phi = 0.0175 \times 0.0175$ and has a length of 230 mm [1]. The electrons or photons are stopped in this material which represents 25.8 radiation lengths, emitting lights that are collected by the avalanche photo-diodes(APDs) in the barrel, and by the vacuum triodes in the endcaps. The observed energy may vary from below 1 GeV up to 1 TeV. After the digitization of the signal, the measured energies are converted to 12 + 2 bit words. The energy itself is digitized to 12 bits, and the 2 extra bits are used to represent the scale factor corresponding to the amplifier that has not been saturated and that has the highest gain. This technique of dynamic range compression is implemented in the Floating-Point Pre-Amplifier (FPPA).

The collisions between the oppositely circulating proton beams at the LHC take place at a rate of 40 MHz. The level 1 trigger rate is supposed to be 10^5 Hz. Taking this L1 trigger rate into account and considering the number of crystals in the ECAL as well as the size of the signal and the number of samplings per signal, we may calculate the amount of information produced by the ECAL which has to be processed by the Upper Level Readout(ULR). This amounts to 1,328 Gigabits per second, which is much higher than the allowed value for the entire CMS event builder, i.e., 500 Gigabits per second. Therefore, the complete readout and storage of the signals from all these crystals are not possible [2]. Only 100 kilobytes are allocated to the ECAL data per event, and we need to find a way to reduce the data to this level.

One way of solving the problem of large data handling is to neglect small signals below a certain threshold. This is called 'zero suppression'. This method allows to remove a significant fraction of the full ECAL data which are within $\pm 1\sigma$ around the pedestal values in the crystals where practically no energy has been deposited. However, when some crystals receive the electrons or photons of sufficiently large energy, we would want to keep the entire information without zero suppression so that we may reconstruct in full detail the physical phenomena occurring in that region.

A compromise between the zero suppression and the full readout has been proposed to reduce the data size to an acceptable level [2] [3]. Application of this Selective Readout(SR) algorithms results in the data sizes of the order of 100 kilobytes, which may be within the data acquisition requirement. The average data size, however, depends on the cuts that are used in the selective readout algorithms. Therefore it is necessary to consider the possibility of further reducing the data size so that the optimization of the cuts can be performed.

The compression of the data can be done in two different ways, say, by lossy and lossless methods. As we want to keep the information as precisely as possible for the selected data, we will limit ourselves to the lossless ones. The commonly used coding methods like the differential coding, the entropy coding, the dynamic coding, the residual parametric coding, and the run length coding fall into this category [4] [5] [6] [7].

In this paper, we report on the investigation of various methods of reducing the ECAL data size. Different algorithms are applied to the binary data of the ECAL signal produced by the CMS simulation code incorporating the Selective Readout, and their compression factors are estimated. In section 2, we describe the Selective Readout criteria that are used in this work. The section 3 details some technical aspects of the event generation using the CMS simulation code. In section 4, we explain various methods of the lossless data compression and estimate the corresponding compression factors. Some algorithms that have been implemented to hardware are also studied and their performances are presented in section 5. The conclusion is given in section 6.

2 Selective Readout

The selective readout, hereafter called SR, may reduce the data size, without too much loss of physical information, at a level allowed by DAQ by identifying the regions in space which contain significant energy. Two types of SR have been developed and their physical and instrumental aspects have been studied thoroughly in the past [3]. One of these is called the tower SR. It is based on the readout of crystal signals in the trigger towers containing a trigger tower which has an energy sum exceeding 1.0 GeV, for instance. The other algorithm called single crystal SR generates the readout information based on the energy of the individual crystal.

In this study, we concentrate on the tower SR, and we consider two possibilities : One(SR1) is the use of two different domains, say, time domain and space domain, according to the size of the transverse energy measured in the trigger towers. If the transverse energy of the tower sum exceeds 2.5 GeV, full ten samplings in time are read out(time domain). For tower energies between 1.0 GeV and 2.5 GeV, only the filtered energy of each crystal is read out instead of the full time samplings(space domain). This method has the advantage of reducing the data size considerably, and it is not difficult to obtain a reduction factor of 50, below the size allowed by the data acquisition system.

The other method(SR2) has only one cut in the transverse energy, say, at 1.0 GeV, and it records the full time samplings. In this case, the data size may become so large that a further data compression stage is needed before they are sent to the DAQ. The availability of many compression algorithms that have already been developed allows this option to be studied in detail. Therefore, we will also estimate the data size in the case of the full readout for various cut values of the transverse energy.

3 Generation of ECAL data using the CMS simulation

In order to test the performances of the compression techniques that will be described in the following sections, we generated fully simulated ECAL data. The full simulation of signal and background events have been done by the CMSIM version 115, with some modification and addition of subroutines when necessary.

First, we incorporated a proposed ECAL endcap trigger tower geometry [8]. The outermost endcap towers overlapping with the last barrel trigger tower are not used. Fig.1 shows the way the crystals are configured in a quarter of the endcap. The mixed energy scale suggested in ref. [2] is taken into account. In the barrel, the minimum value of LSB is chosen to be 20 MeV, at η=0. The LSB in energy follows $sin^{-1}\theta$ distribution, whereas it stays constant in the transverse energy scale. In the endcaps, the LSB in energy is fixed, instead. The variation of electronics noise level which has a similar behaviour is set by default in the CMSIM code, as can be seen in Fig.2(a)-(b). However, we made a modification of the ecal.tz file to remove the zero suppression which is applied at 1σ by default. Also, the noise is generated independently for each time sampling. The pedestals are set to 25 ADC counts, which corresponds to 500 MeV at η=0. In Fig.3(a)-(d) we plotted the distribution of the crystal energies, the ADC counts, the ADC counts in the barrel only, and the ADC counts in the endcaps only, respectively, for all the sampling values of an event without any signal, i.e., only with noise. The noise levels are set at E_t=30 MeV and E=150 MeV in the barrel and in the endcaps, respectively. The modeling of the signal is done using the formula proposed in ref. [9], which reads

$$f(t) = (1 + \frac{t - tof - \alpha\beta}{\alpha\beta})^{\alpha} \cdot e^{-\frac{(t - tof - \alpha\beta)}{\beta}}, \quad if \quad t > tof - \alpha\beta$$

$$0, \qquad\qquad\qquad\qquad otherwise$$

, where α=1.5, β^{-1}=0.568, and tof represents the time of flight of the hit.

The QCD events with the transverse momentum above 100 GeV/c and the minimum bias events have been generated. On top of each QCD event, 20 minimuim bias events have been piled-up. This corresponds to the luminosity of 10^{34} cm^{-2}s^{-1}, approximately. The pile-up of events coming from the interactions of the bunch crossing at different times has not been included assuming that the starting time of the shower which gives rise to the signal measured at a given time can be known in some way. (This can be achieved by a filter that determines both the jitter and the size of the signal from each crystal) Fig.4(a)-(d) represent the crystal energy, E_t, in the barrel, and the crystal energy in the two endcaps, respectively, for a Higgs event with four electrons in the final state.

In the barrel, the crystals are grouped into 5 by 5 matrices. The position of a matrix is represented by two indices I_η and I_ϕ. The ranges of I_η and I_ϕ are given by 12-45 and 1-72, respectively. The geometry of endcaps requires a special indexing scheme. Following the most recent proposal, the I_η takes on the values from 5 to 11, and from 46 to 52 for the forward and backward endcaps, respectively. The distribution of E_t in the trigger towers for the same event is shown in Fig.5(a). Fig.5(b) shows the positions in I_ϕ vs. I_η plane of the towers that have recorded more than 1 GeV of transverse energy, as well as the trigger towers adjacent to them.

The generated QCD events have been used to estimate the occupancy of the towers and the results are given in Table 1 for different SR criteria. Also shown in Table 1 are the size of the events. In the case of the SR2, where all ten time sampling are read out, four different cuts are considered; E_t >2.5 GeV, E_t >1.0 GeV, E_t >0.5GeV and E_t >0.3 GeV. The coarse grain data of about 4 kbytes needs to be added to each of them.

The ULR scheme of ref. [3] suggests that the crystals corresponding to a supermodule are read out by one ULR crate, thereby necessitating 50 ULR crates in all. The compression of the ECAL data is supposed to be performed in the Data Concentrator Card(DCC) which collects data from the ULR cards and passes it to the DAQ system. Therefore, it is required that our simulated events be splitted into different supermodules. We have thus produced the event files corresponding to the 50 supermodules.

The size of the data obtained by applying the SR1 is well below 100 kilobytes which is allowed by DAQ. In the case of SR1, it is required that the filtered energy of the crystals be estimated with a good precision so that the physics of interest in CMS is not affected significantly by keeping only the filtered value instead of the full time sampling values. Also,

SR type	no. of towers	no. of crystals	event size
SR1(time + space)	68(262)	1,482(5,416)	41.2 kB
SR2-1(time, E_t >2.5 GeV)	68	1,482	29.8 kB
SR2-2(time, E_t >1.0 GeV)	330	6,898	138.9 kB
SR2-3(time, E_t >0.5 GeV)	908	18,936	381.4 kB
SR2-4(time, E_t >0.3 GeV)	1,554	32,935	662.8 kB

Table 1: *Occupancies of the towers and the crystals. Two types of the selective readout are compared. In the case of SR1, the values in the parentheses correspond to the space domain data. In the case of SR2, where the space domain is not used, three different cuts on E_t are considered. An event is composed of a high-P_t QCD event piled-up with 20 minimum bias events. One hundred of such events have been used. The coarse grain data of about 4 kbytes needs to be added to each of them.*

the time needed for calculation must be very short, say, less than the L1 trigger latency of several microseconds. In the case of SR2, the increase of the data size in lowering the threshold makes it necessary to consider a compression of the data to a lower level that can be processed by DAQ.

4 Algorithms for the lossless data compression and Estimation of the compression factors

In this section we introduce five types of data compression algorithms which are most commonly used in the communication systems or in archiving the data files of the computers. We apply these methods to the ECAL data and evaluate the corresponding compression factors.

4.1 Differential coding

The signal from a given crystal is shaped by the preamplifier so that a decaying time of about 300 ns is introduced. A sampling is performed every 25 ns and the measured voltage is digitized by a 12-bit ADC after passing through the floating point unit(FPU) which determines the dynamic range of the output signal. Fig.6(a) shows the typical shape of the signal and the points of samplings in unit of 25 ns. The hights of the signal at the sampling points are recorded.

Another possible way of keeping the same amount of information is to record the value at the first sampling point and then to record the differences between the first and the second, between the second and the third, and so on. This method is called the differential coding. The advantage of such a coding scheme is that the numbers to record are usually smaller than the standard coding, and the number of bits needed to record the numbers may be smaller even though we need to introduce one more bit that represents the sign of the differences which can be negative. Fig.6(b) shows the differences between two neighboring samples.

This simple method allows to reduce the length of the data, especially if the signal varies slowly with respect to the sampling interval. In our case, however, the rise and fall of the signal is so fast that no significant gain in the data length may be expected. Nevertheless, it will be instructive to do an exercise and estimate the compression rate using the data of Fig.6(a),(b). The number of bits needed to code the sampled value x_i is given by $Int(log_2(x_i)+1)$, whereas the difference between the neighboring samples $d_i = x_i - x_{i+1}$ can be coded by $Int(log_2(abs(d_i+0.5)+1)+1)+1$ bits. These numbers of bits corresponding to the 14 sampling values and also to the differences are given in Table 2. The maximum number of bits in the two cases are 12 and 11, respectively, and no significant gain is achieved by applying this algorithm.

Samplings	1	2	3	4	5	6	7	8	9	10	11	12	13	14
Normal coding	7	11	11	11	12	11	11	11	10	10	9	8	8	7
Differential coding	11	11	9	7	9	10	10	10	10	9	9	8	8	-

Table 2: *Number of bits needed to record the differences between the neighboring sample values*

4.2 Residual parametric coding

If the shape of the signal has a time dependence that can be approximated by a simple function, and if the difference between the signal and the function is small in most cases, we may reduce the data size by coding the differences. To apply this algorithm called residual parametric coding, we proceed as follows :

(1) Generate a model as shown in Fig.6(c)

(2) Normalize the model to the signal so that the maxima of the two have the same magnitudes. See Fig.6(d)

(3) Calculate the differences between the signal and the model at the sampling points.

(4) Make the data file which has the structure described below.

The data file starts with the header that represents the maximum bit length, N_b, of the difference values. The header is then followed by the value of the signal at its maximum, and then by the difference values having N_b bits each. Using the data and the model function shown in Fig.6(c) and Fig.6(d), we obtain the Table 3. As we have 14 samples of 12 bits, the initial length is 168 bits. With this residual parametric coding, only $100(= 4 + 12 + 14 \times 6)$ bits are needed, giving a compression factor of 1.68.

Samplings	1	2	3	4	5	6	7	8	9	10	11	12	13	14
Normal coding	12	12	12	12	12	12	12	12	12	12	12	12	12	12
R.P.C.	2	4	4	5	6	5	3	5	1	6	3	3	5	5

Table 3: *Number of bits needed to record the differences between the signal and the model*

The signal that we used for this exercise is relatively well represented by the model function. In the case of the real signal from the test modules, however, the shape of the signal changes from event to event, and also a time jitter is present. The resulting residuals are therefore so large that no significant gain is obtained by this method in real application.

From the above two investigations, we see that data compression in the time frame of the signal from the individual crystal is not efficient. Therefore we will consider other methods which are statistical in nature. Those methods are based on the dynamic assignments of the data length according to the frequency of occurrence of a given number or a given set of numbers. Huffman coding method is an example of the former, and the dictionary methods fall into the latter category. We study the principle and performance of these two methods in the following. Another simpler dynamic coding method is also suggested and its efficiency evaluated.

4.3 Huffman coding

When the same numbers appear repeatedly, we may replace these numbers with some other numbers having shorter number of bits. The best reduction of the total length of a set of numbers can be obtained by associating more frequently appearing numbers to shorter numbers. Huffman's coding method gives the optimized assignment rule which is uniquely decodable [10]. We take an example of Huffman coding to illustrate this method. Let us consider a set of 6 numbers $A_i, (1 \leq i \leq 6)$ occurring with the probabilities 0.4, 0.3, 0.1, 0.1, 0.06, 0.04, respectively. The corresponding Huffman codes are given in Table.4.

Number	Probability	code
A_1	0.4	1
A_2	0.3	00
A_3	0.1	011
A_4	0.1	0100
A_5	0.06	01010
A_6	0.04	01011

Table 4: *An example of Huffman code assignment*

The average length of this code is $0.4\times1+0.3\times2+0.1\times3+0.1\times4+0.06\times5+0.04\times5 = 2.2$ bits/number while the fixed length coding requires at least 3 bits for each number. Once the code is fixed, the coding and decoding are done in a unique way. The numbers are written in a form of a block code. For example, the sting 01000101001100 is unambiguously decoded as $A_4A_5A_3A_2$.

To apply the Huffman coding method to the ECAL data, we used the distribution of the ADC counts shown in Fig.7(a). The shortest Huffman code is associated to the most frequently occurring value IP. To the ADC values from IP-7 to IP+7 are assigned the codes with its length varying from 2 to 9 bits. For the values smaller than IP-7, or larger than

IP+7, a 4+16 bits structure is used, where the Huffman code of 4 bits (0000) is followed by the full 16 bits of the ADC. The Huffman code assignment is given in Table 5.

ADC counts	Probability %	code	word length
IP-8 or smaller	0.19	0000	4+16
IP-7	0.22	101101001	9
IP-6	0.42	10110101	8
IP-5	0.78	1010101	7
IP-4	1.72	101100	6
IP-3	4.42	10111	5
IP-2	10.10	001	3
IP-1	17.40	110	3
IP	21.05	01	2
IP+1	17.69	111	3
IP+2	10.74	100	3
IP+3	5.19	0001	4
IP+4	2.53	10100	5
IP+5	1.46	101011	6
IP+6	0.95	1011011	7
IP+7	0.67	1010100	7
IP+8 or larger	4.46	0000	4+16

Table 5: *Huffman code assignment for the ECAL ADC values, Here IP respresents the ADC value corresponding to the pedestal, which is 25 in this case.*

Table 6 shows the performance of the Huffman coding for the 100 hard QCD events with five different SR types. A good compression is obtained with this *truncated* Huffman method. Also shown in the same table is the compression factors observed by applying the unix command *compact* that uses an adaptive Huffman code.

4.4 Dictionary method

The dictionary method uses the property of many data types that contain repeating code sequences. It can be divided into two main groups which are based on the algorithm developed and published by A. Lempel and J. Ziv [11] . The point of the first group is to try to find if the character sequence currently being compressed has already occurred earlier in the input data. In the case the same sequence is found, instead of repeating it, the algorithm outputs a pointer. See Fig.8(a). The second group creates a dictionary of the phrases that occur in the input data. When they encounter a phrase already present in the dictionary, they just output the index number of the phrase in the dictionary as shown in Fig.8(b).

In most of the unix machines, the commands *compress* and *gzip* performs the compression using the dictionary method. Therefore, we have evaluated the compression rates using the

SR type	event size	Huffman coding	*compact*
SR1(time + space)	41.2 kB	10.4 kB(4.1)	13.7 kB(3.0)
SR2-1(time, E_t >2.5 GeV)	29.8 kB	7.6 kB(3.9)	9.5 kB(3.1)
SR2-2(time, E_t >1.0 GeV)	138.9 kB	33.0 kB(4.2)	49.0 kB(2.8)
SR2-3(time, E_t >0.5 GeV)	381.4 kB	87.0 kB(4.3)	129.8 kB(2.9)
SR2-4(time, E_t >0.3 GeV)	662.8 kB	148.0 kB(4.4)	221.0 kB(3.0)

Table 6: *The event sizes with and without Huffman coding. Results with the Truncated Huffman coding that uses a fixed table and the Adaptive Huffman coding with a variable table are shown.*

SR type	event size	*compress*	*gzip*
SR1(time + space)	41.2 kB	11.6 kB(3.5)	12.0 kB(3.4)
SR2-1(time, E_t >2.5 GeV)	29.8 kB	8.0 kB(3.7)	8.3 kB(3.6)
SR2-2(time, E_t >1.0 GeV)	138.9 kB	38.9 kB(3.5)	41.5 kB(3.3)
SR2-3(time, E_t >0.5 GeV)	381.4 kB	99.7 kB(3.8)	106.8 kB(3.6)
SR2-4(time, E_t >0.3 GeV)	662.8 kB	168.6 kB(3.9)	184.0 kB(3.6)

Table 7: *The compression factors for the ECAL data using the unix commands*

simulated ECAL data in the two cases. The results that we obtain for different SR criteria are summarized in Table 7.

4.5 Dynamic coding

It was suggested by Busson et al. [12] that a reduction of the data size can be done by simply choosing the word length between one byte and two bytes. In this dynamic coding scheme, the first one or two bits of the 8 bits is used to indicate the length of the signal from each crystal. A slightly modified data structure is described in the following.

The energies of the 25 crystals in a given trigger tower are stored consecutively. The 10 values corresponding to the ten time samplings for the first crystal are written first, followed by the 10 time samplings of the next crystal, etc. By allocating one byte(eight bits) to each energy value, a minimum of 250 bytes are needed to record all the energy values. If a crystal has an energy exceeding the maximum that can be represented by the eight bit, one or two more bytes are used. The number of crystals that need two or three bytes and their sequential number are also coded in the data train. These numbers can be represented by a one byte word. We suggest the data structure as described below.

(1) In the first byte, the first bit(MSB) is used to specify the data type: either full ten time samplings or the filtered value. The next bit is reserved to flag the presence of very large signal which needs to be coded by 3 bytes. The remaining 6 bits are used to assign the position in η of the corresponding trigger tower which varies from 1 to 56.

(2) The first bit of the next byte indicates the presence of the 2- byte data, and the following seven bits specify the position in ϕ ranging from 1 to 72.

(3) In the case that 2-byte data are present, the number of the crystals which recorded such signals is written in the next byte. Let's call it $N_2(0 \leq N_2 \leq 250)$. This byte can be followed by another byte, which is, N_3, if necessary.

(4) The sequential numbers of the crystals that need 2-byte recording occupy N_2 bytes. Sometimes it can be followed by N_3 addresses for crystals having very high energy deposit.

(5) Total of $250 + N_2 + N_3 * 2$ bytes to record the ADC counts of a trigger tower chosen by the SR.

The event sizes with and without dynamic coding are given in Table 8 for various SR parameters. About a factor of two compression is obtained, as expected.

SR type	event size	dynamic coding	compression factor
SR1(time + space)	41.2 kB	21.0 kB	1.96
SR2-1(time, E_t >2.5 GeV)	29.8 kB	15.0 kB	1.98
SR2-2(time, E_t >1.0 GeV)	138.9 kB	69.8 kB	1.99
SR2-3(time, E_t >0.5 GeV)	381.4 kB	191.7 kB	1.98
SR2-4(time, E_t >0.3 GeV)	662.8 kB	333.0 kB	1.99

Table 8: *The event sizes with and without dynamic coding.*

5 Commercially available devices for compression

Some commercial devices that perfom the lossless data compression are found to be available in the form of integrated circuits. The evaluation softwares for those products have been used to see the possibility of using such devices for our purposes.

5.1 ALDC

First, we considered the Adaptive Lossless Data Compression(ALDC) algorithm which is running in an IBM product ALDC1-40S-M [13]. We ran the ALDC software on the data files generated with and without dynamic coding. Table 9 shows the compression factors.

In order to check that the ALDC algorithm is efficient when applied to a single 3 by 3 tower array, we have generated a single electron with different η and P_t values and passed them through the full detector simulation. The original and the compressed data sizes are given in Table 10. The compression factor remains to be more than 2 in almost all cases. The variaion of the compression factor in η and P_t is consistent with our expectation taking into account the noise level in barrel and in endcaps, and the size of the electromagnetic shower.

SR type	event size	size with ALDC	compression factor
SR1(time + space)	41.2 (21.0) kB	15.6 (11.8) kB	2.6 (1.8)
SR2-1(time only, E_t >2.5 GeV)	29.8 (15.0) kB	11.7 (8.5) kB	2.7 (1.7)
SR2-2(time only, E_t >1.0 GeV)	138.9 (69.8) kB	53.1 (40.4) kB	2.6 (1.7)
SR2-3(time only, E_t >0.5 GeV)	381.4 (191.7) kB	139.4 (109.5) kB	2.7 (1.7)
SR2-4(time only, E_t >0.3 GeV)	662.8 (333.0) kB	245.1 (189.9) kB	2.7 (1.8)

Table 9: *The data size before and after applying the ALDC algorithm and the compression ratios with various E_t thresholds. The values in parentheses correspond to the dynamically coded data.*

$P_t(GeV)$	η=0.1	η=1.0	η=1.6	η=2.0
5.0	2.93(1.87)	2.91(1.86)	2.19(1.44)	2.11(1.37)
10.0	2.87(1.80)	2.84(1.85)	2.19(1.43)	2.04(1.34)
50.0	2.79(1.77)	2.84(1.85)	2.18(1.42)	1.98(1.28)

Table 10: *The compression factor using ALDC for different P_t and η values of a single electron. The data corresponds to a 3 by 3 tower array. The values in parentheses are the compression ratios for the dynamically coded data.*

SR type	event size	size with DCLZ	compression factor
SR1(time + space)	41.2 (21.0) kB	12.1(10.9) kB	3.3 (1.9)
SR2-1(time only, E_t >2.5 GeV)	29.8 (15.0) kB	8.8 (7.7) kB	3.4 (2.0)
SR2-2(time only, E_t >1.0 GeV)	138.9 (69.8) kB	42.1 (34.6) kB	3.3 (2.0)
SR2-3(time only, E_t >0.5 GeV)	381.4 (191.7) kB	110.6 (90.8) kB	3.4 (2.1)
SR2-4(time only, E_t >0.3 GeV)	662.8 (333.0) kB	193.6 (155.7) kB	3.4 (2.1)

Table 11: *The data size before and after applying the DCLZ algorithm and the compression ratios with various E_t thresholds. The values in parentheses correspond to the dynamically coded data.*

5.2 DCLZ

Another device that we found in the market is called Data Compression Lempel Ziv(DCLZ) [14]. Similar checks have been done with the software, and the results are given in Table 11.

6 Conclusion

We have studied the methods for CMS-ECAL data reduction using the fully simulated QCD events with some modification of the CMSIM package. A comparison of the compression factors that result from different methods is given in Table 12. One hundred SR2-2 events have been chosen to get this table. We get similar results when other SR criteria are used. The values quoted in the parentheses correspond to the case where the compression is applied not to the entire event but to the smaller data files from the supermodules. From this result, we observe that a compression factor of at least 2 can be achieved. The Huffman method with fixed table gives the best compression factor for our ECAL data.

Type	coding method	compression factor
Fixed table	Huffman	4.2(4.0)
Variable table	Huffman	3.1(3.0)
compress	dictionary	3.6(3.3)
gzip	dictionary	3.3(3.2)
dynamic	8 or 16 bits	2.0(2.0)
ALDC	dictionary	2.6(2.6)
DCLZ	dictionary	3.3(3.2)

Table 12: *The compression factors obtained by various types of algorithms. The values in the parentheses correspond to the case where the compression is applied to the data from each supermodule.*

7 Acknowledgement

The authors are grateful to Claude Charlot and Ludwig Dobrzynski for their helpful discussions and comments. This work has been supported by the international cooperation program of the Korea Science and Engineering Foundation (KOSEF grant no. 985-0200-012-2) and the Centre National pour la Recherches Scientifiques of France.

References

[1] The Electromagnetic Calorimeter Project, Technical Design Report, CERN/LHCC 97-33, CMS TDR 4 (1997)

[2] R. Benetta et al., ECAL Data Volume, CMS note 1997/059.

[3] J.C. Silva et al., CMS ECAL Data Concentrator-System Design Description, CMS note 1999/012.

[4] P. Plume, Compression des données, Eyrolles, 1993

[5] J.A. Storer, Data Compression Methods and Theory, Computer Science Press, 1988

[6] IBM J. Res. Develop. Vol.42 No.6, November, 1998

[7] D. Salomon, Data Compression, The complete reference, Springer (1998)

[8] W. Badgett, Trigger Tower Definition Issues, Presented in the TriDAS Meeting of June 16, 1998.

[9] Ph. Busson, Amplitude and time measurement of ECAL signals with optimum digital signal processing, presented in the CMS ECAL readout meeting of May 25, 1998.

[10] R.C. Gonzalez and R.E. Woods, Digital Image Processing, Addison Wesley (1993)

[11] J. Ziv and A. Lempel, A Universal Algoritm for Sequential Data Compression, IEEE Trans. Info. Theory IT-23. No. 3. (1997) 337-343

[12] Ph. Busson, A. Karar, and G.B. Kim, Study of ECAL data compression, presented in the CMS ECAL readout meeting of May 6, 1998.

[13] ALDC1-40S-M Data Sheet Manual, IBM Document Number : DCAL40DSU-02, November 2, 1994.

[14] For further informations, see http://www.aha.com

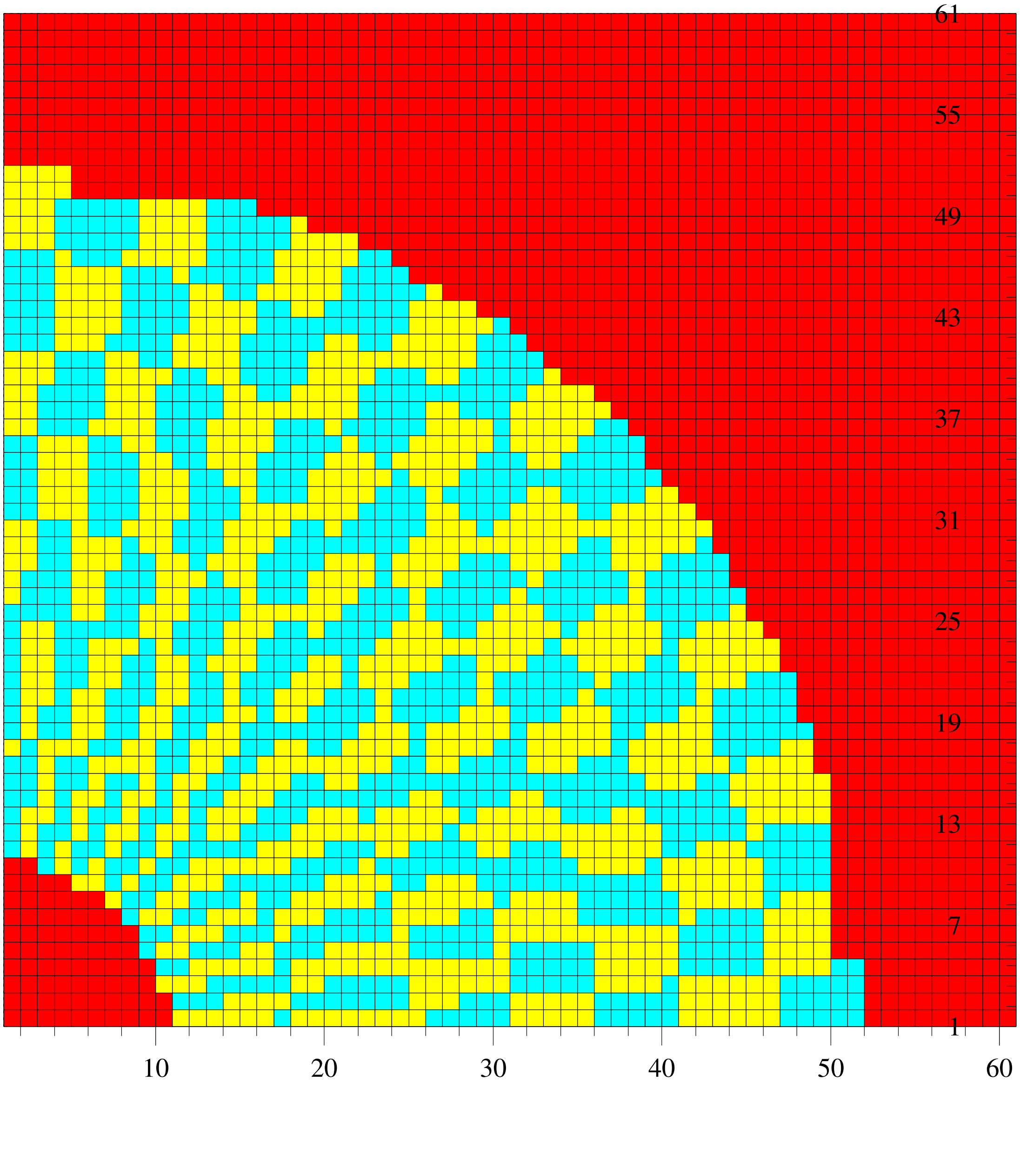
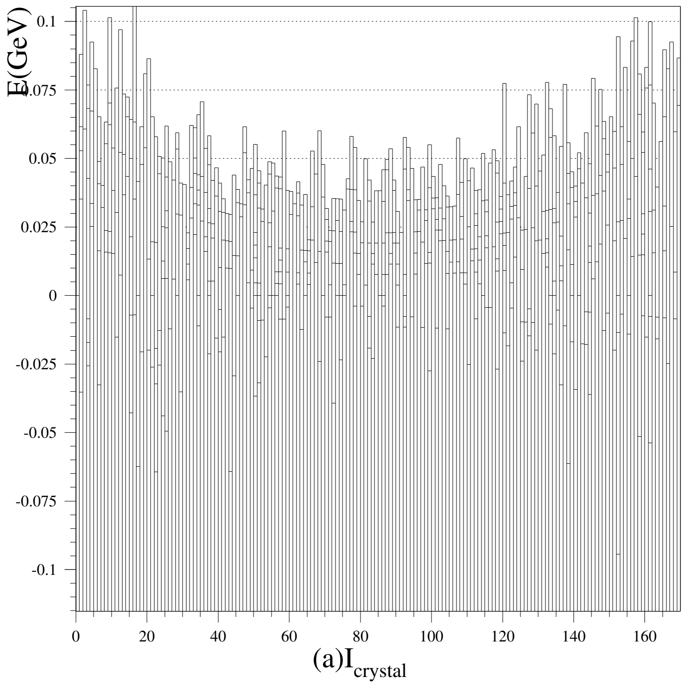


Figure 1: *ECAL endcap trigger tower geometry*



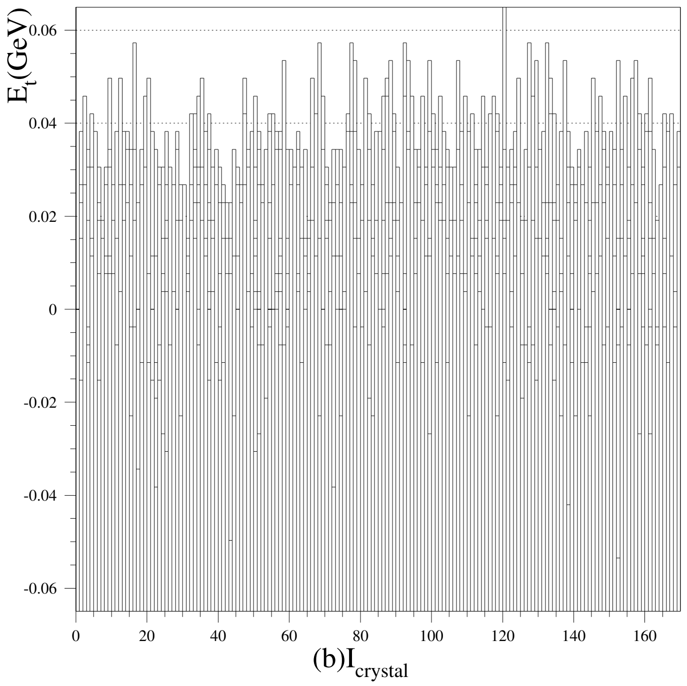


Figure 2: *(a)Noise level in E for barrel vs. η, (b)Noise level in E_t for endcaps vs. η.*

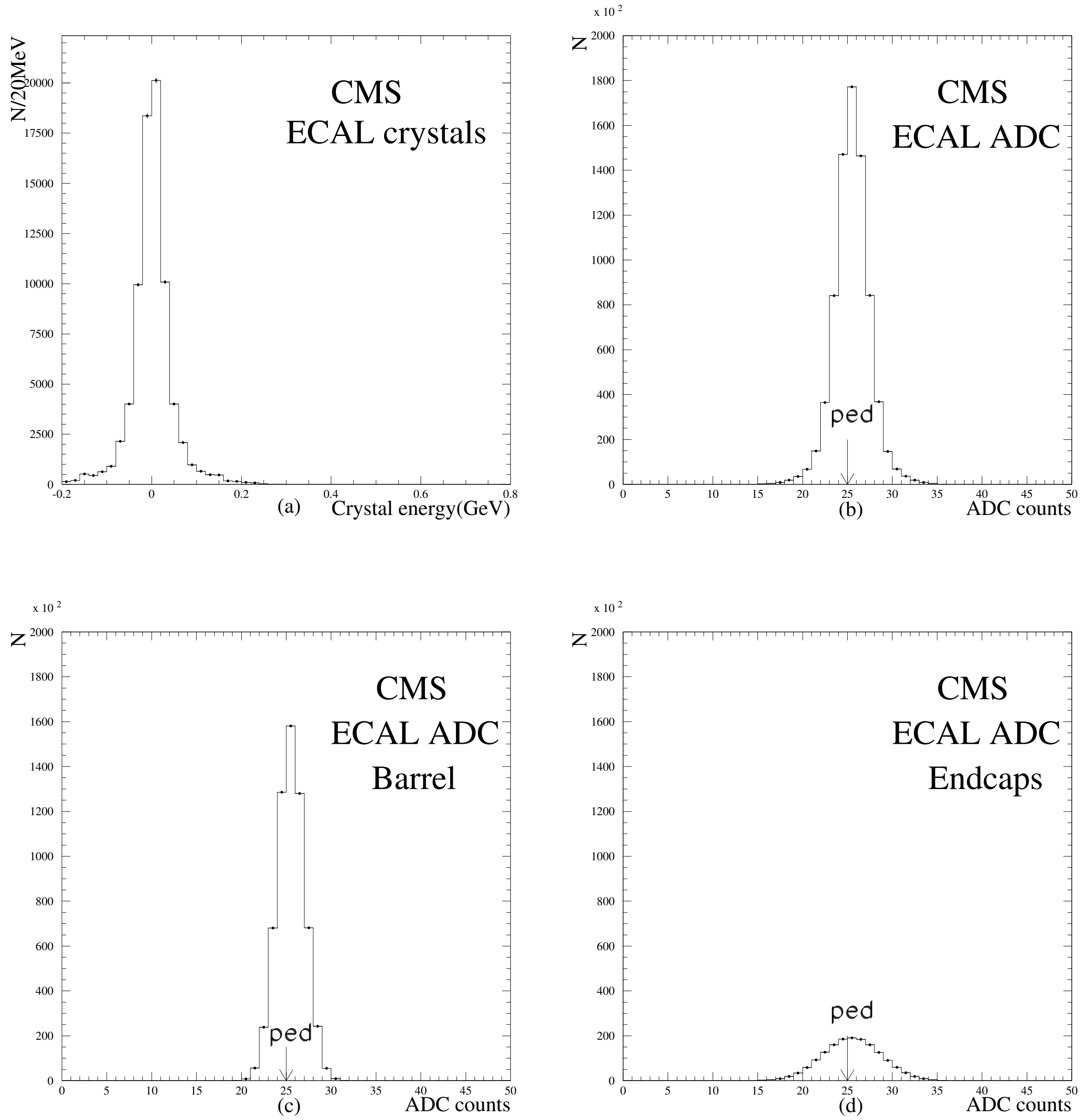


Figure 3: *(a)Distribution of the crystal energy, (b)The ADC counts, (c)The ADC counts in the barrel, and (d)The ADC counts in the endcaps, for all the samplings in an event containing no signal .*

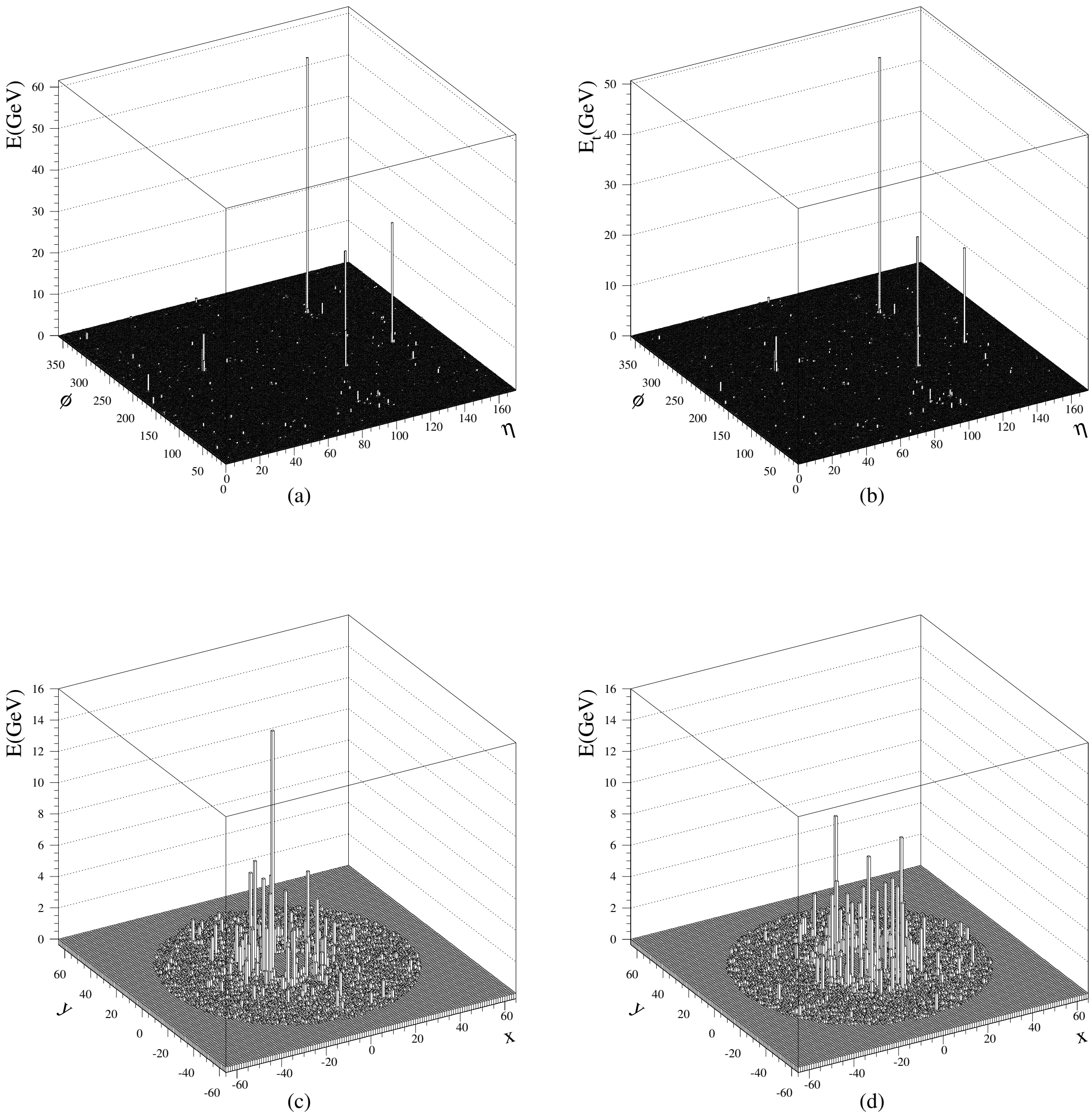
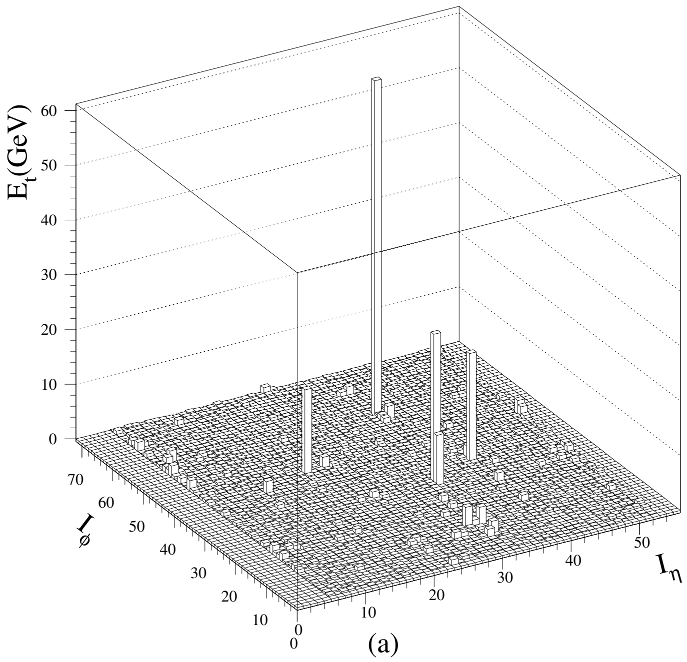
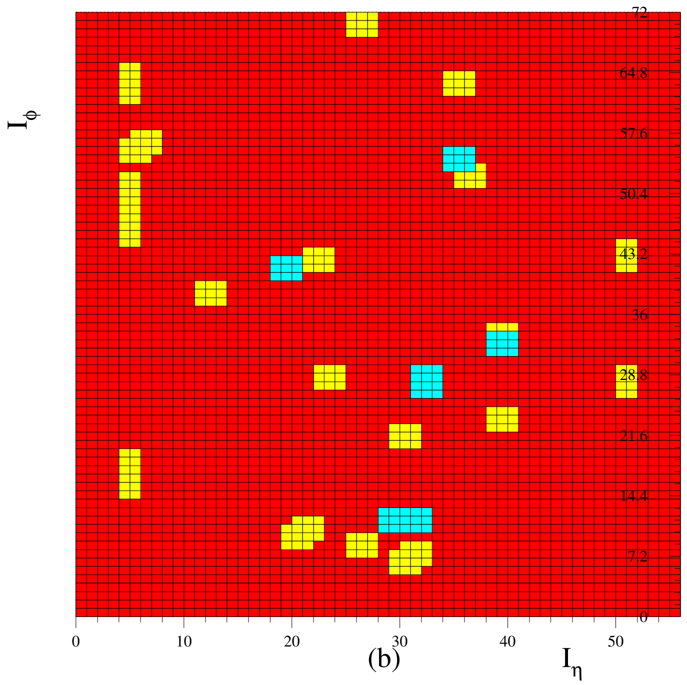


Figure 4: *(a)The crystal energy in the barrel, (b)the crystal E_t in the barrel, (c)The crystal energy in the forward endcap, and (d)the crystal energy in the backward endcap. A Higgs particle of 150 GeV/c^2 decays into four electrons .*



(a)



(b)

Figure 5: *(a)The E_t distribution of the trigger towers for the Higgs event, (b)The positions in I_ϕ vs I_η plane of the towers that have recorded more than 1 GeV of transverse energy, as well as the adjacent trigger towers. The gray cells correspond to the time domain, and the brighter cells correspond to the space domain.*

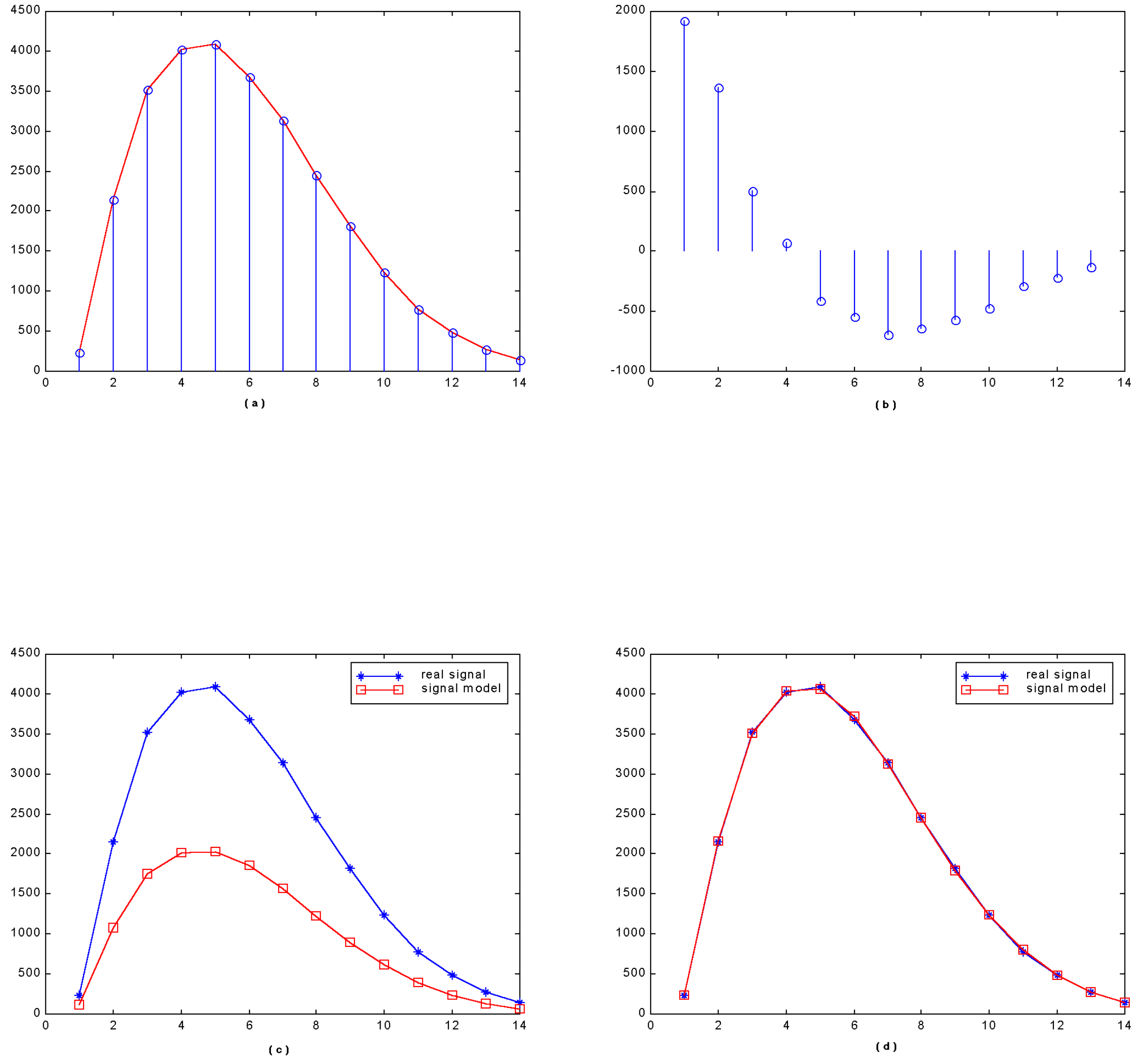


Figure 6: *Differential coding: (a)Shape of the signal vs. time, (b)Differences between neighboring samples, and Residual parametric coding: (c)Signal and Model before normalization, (d)Signal and Model after normalization*

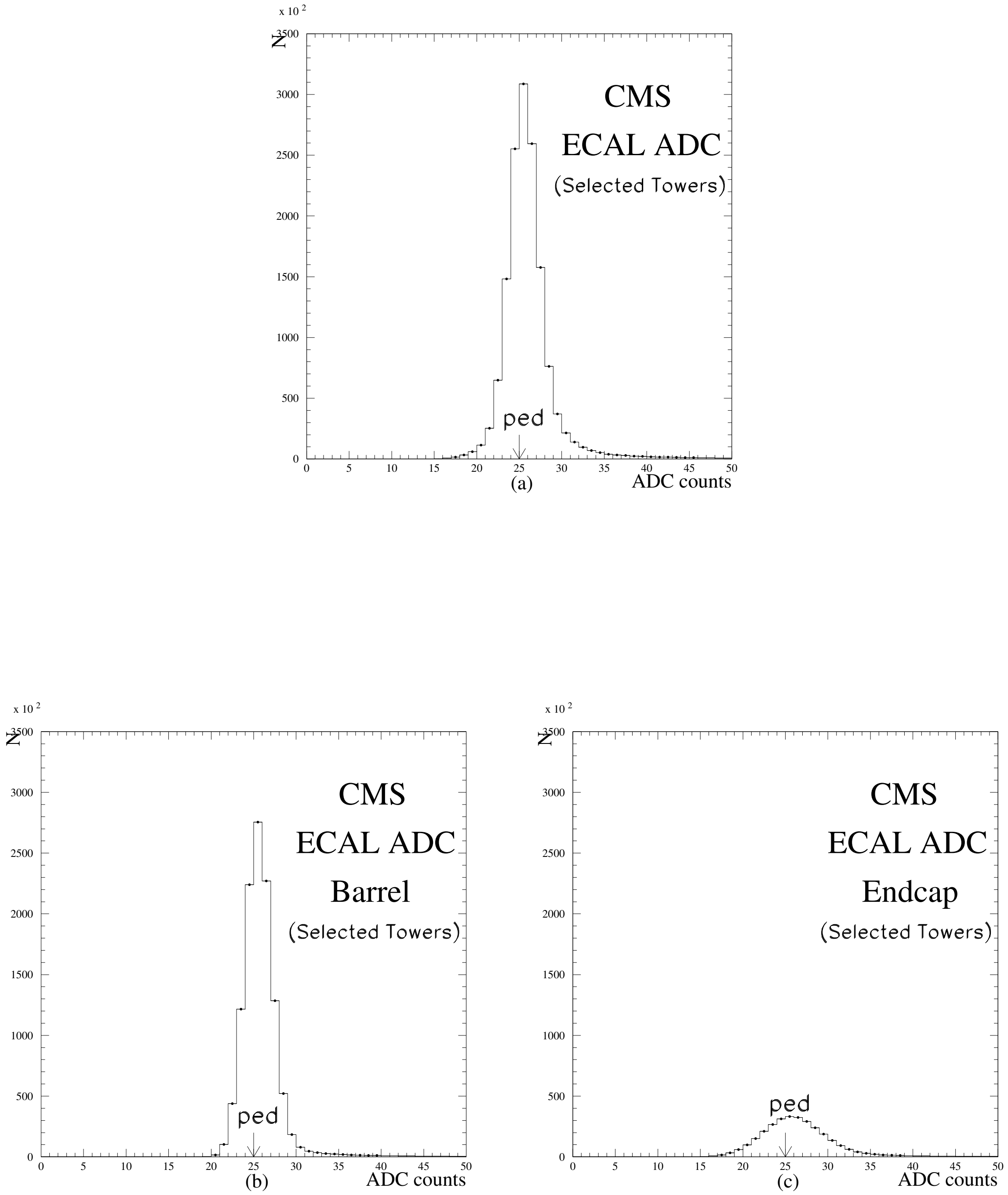


Figure 7: *(a)Distribution of the ADC counts used to generate the Huffman codes, (b)Same distribution with barrel, (c)Same distribution with endcaps.*

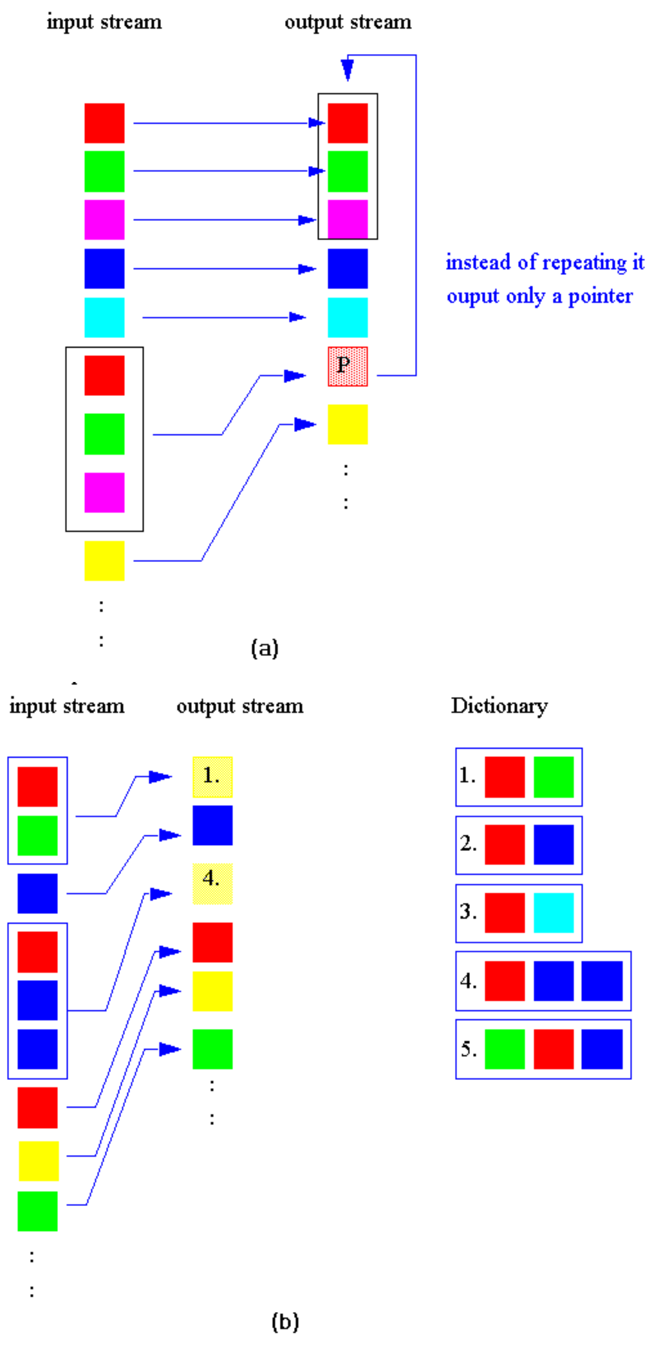


Figure 8: *Dictionary methods : (a)scheme 1 and (b)scheme 2*