An Improved Algorithm for On-Chip Clustering and Lossless Data Compression of HL-LHC Pixel Hits

An Improved Algorithm for On-Chip Clustering and Lossless Data Compression of HL-LHC Pixel Hits

Giuseppe Baruffa, Pisana Placidi, *Member, IEEE,* Andrea Di Salvo, Sara Marconi, and Andrea Paternò

***Abstract*—A prototype chip, called RD53A, has been designed by the RD53 collaboration to face the very high hit and trigger rate requirements (up to 3 GHz/cm^2 and 1 MHz, respectively) of the High Luminosity LHC experiment upgrades. In this paper, an improved algorithm for data compression, capable of sustaining the very high data volume and proposed to be implemented in the periphery of the chip, is presented: it exploits Run Length Encoding (RLE) and Variable Length Coding (VLC) to compact chip pixel hit patterns. The compression and decompression algorithms are implemented with MATLAB, and the performance is calculated taking into account the RD53A data readout implementation and its chip simulation and verification framework (called VEPIX53). In all considered cases, the results show that the RLE and VLC combination achieves a data compression ratio between 1.57 and 1.62, resulting in a bitstream size reduction between 36.2% and 38.4% with respect to the rate of the current data transmission format.**

I. Introduction

A PIXEL readout chip has been designed within the RD53 Collaboration to face the challenging requirements of the High Luminosity LHC (HL-LHC) experiment upgrades, called RD53A [1]. The chip has to sustain very high hit and trigger rates (up to 3 GHz/cm^2 and 1 MHz, respectively), together with an high radiation level. Due to the increase in data rates, on-chip lossless data compression should be considered to reduce the required bandwidth in the development of final pixel chips for the experiments [2]. In addition, low power consumption and strict requirements on the complexity of the algorithm implementation should be taken into account.

The bidimensional nature of pixel hit patterns and the associated charge information introduce significant complexity on the efficiency estimation of the readout binary encoding, without knowing all the details of the detector, as highlighted in [2]. For instance, the readout encoding used by the ATLAS FE-I4 chip shows that data compression can achieve a 40% reduction of the total readout bandwidth [2]. In [3], two on-chip algorithms exploiting arithmetic and Huffman coding are proposed for HL-LHC. However, the performance analysis has been carried out by considering a full GEANT4 simulation with only 2 GHz/cm^2 hit rate, neglecting constraints due to the RD53A prototype implementation.

The RD53A sensitive area is arranged as an array of 192×400 pixels, each one of size 50 μm × 50 μm, with two different digital buffering architectures. The sensitive area is placed at the top of the chip and, in the considered digital buffering scheme, the pixel matrix is built up of 8×8 pixel cores; each core is logically divided into 16 four-pixel regions. Thus, each region spans horizontally by four columns and vertically by one row. The content of regions enclosing at least one pixel hit is copied into FIFOs placed at the bottom of each core-column. The chip output is encoded with the Aurora 64b/66b protocol over 1 to 4 parallel lanes [1], each nominally capable to convey up to 1.28 Gb/s.

In this work, an improved algorithm for data compression capable of handling the high data volume has been considered for implementation in the chip periphery: it exploits *Run Length Encoding* (RLE) and *Variable Length Coding* (VLC). Additionally, Huffman coding has been taken into account. The performance of the proposed algorithms has been investigated in MATLAB by using the output produced by the VEPIX53 chip analysis and simulation framework [4] (see Fig. 1), thus providing the possibility of importing hit patterns from physics data of the ATLAS and CMS experiments. The data analysis simulation chains of the experiments use ROOT, a common tool developed in C++, to flexibly obtain statistics on the events [4]. The format used by ROOT for representing big data sets is the so-called *TTree*. The hit generator is written in SystemVerilog (SV), from which, by using Direct Programming Interface (DPI), it is possible to set up interfaces with C++. Functions can be therefore defined in the SV hit generator, which directly calls the ROOT routines. To test the performance of the algorithm, the Monte Carlo data from the CMS data analysis framework (CMSSW), including events related to the layer 0 of the pixel detector, are combined with constrained random input data and with meaningful cluster distribution (producing stimuli with the expected hit rate of 3 GHz/cm^2). In particular, the modules at the center of the barrel, with a pixel size of 50 μm × 50 μm, sensor thickness of 150 μm, digitizer threshold of 1500 e$^-$ and a pile-up of 140, are considered.

II. Proposed Data Compression Algorithm

In this scenario, the redundancy in the generated chip output data can be reduced by exploiting data compression algorithms. In Fig. 1, where an example of the chip output for

Manuscript received November 15, 2018. This work was supported by INFN ("Chipix65 collaboration") and by "Fondo di Ricerca di Base 2018", funded by the University of Perugia (project no. PJ RICBA18PP).

G. Baruffa is with Università degli Studi di Perugia, Perugia, Italy.

P. Placidi is with Università degli Studi di Perugia and with Istituto Nazionale di Fisica Nucleare (INFN), Perugia, Italy (e-mail: pisana.placidi@unipg.it).

S. Marconi is with CERN, Geneva, Switzerland.

A. Di Salvo and A. Paternò are with Istituto Nazionale di Fisica Nucleare (INFN), Turin, Italy and Politecnico di Torino, Turin, Italy.

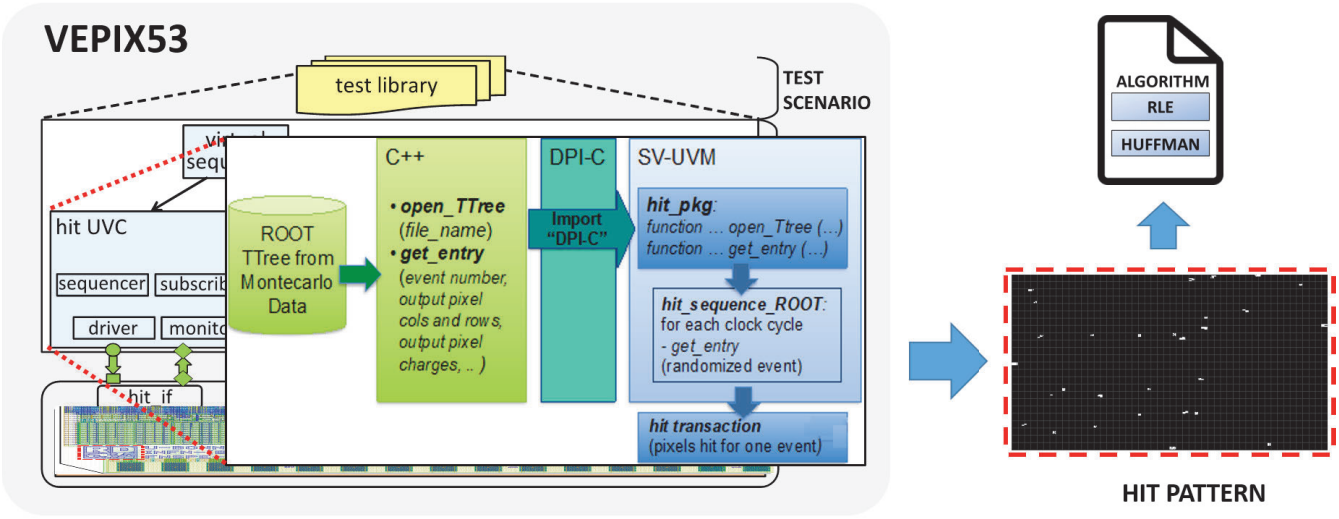

Fig. 1. Implemented simulation workflow.

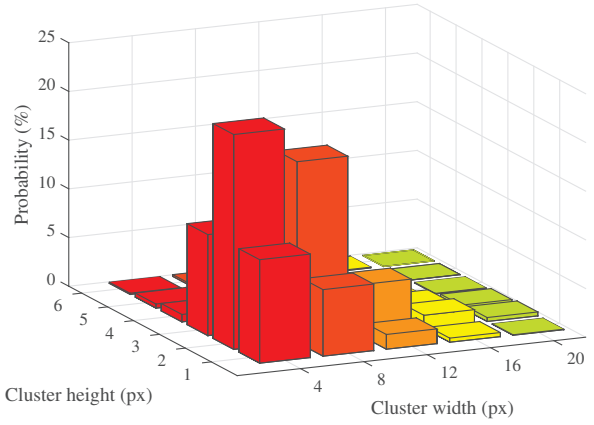

Fig. 2. Occurrence of cluster sizes.

Fig. 3. Traditional encoding of the pixel values in active regions.

a single trigger event hit pattern is reported, the white areas represent active pixels (i.e., with a non-null reading). The hit pattern is sparse, with large blank areas including only few small clusters of active pixels, as highlighted by the cluster size histogram of Fig. 2. The histogram shows that the typical hit cluster size is not larger than 3×12 pixels. In the chip, a smart encoding of the pixel hits (denoted as *current* in the following) is already implemented at the core-column level, after the trigger selection. Therefore, only the active regions (regions including at least one active pixel for the specified trigger) address and content are encapsulated in the Aurora packets: 7 bits are used for the column address, 9 bits for the row address, and 16 bits for the readings of the Time-over-Threshold (ToT) values (Fig. 3). Although this packetization protocol is efficient in dealing with the sparsity of the pixel matrix data, it does not account for the spatial correlation among active regions and for the non-uniform statistical distribution of the pixel values. To address this inefficiency, a customized data compression algorithm, exploiting RLE and VLC [5], is considered in the following.

In the algorithm, instead of including the address for every active region, we propose to use a single address for a cluster of regions. Consecutive inactive regions within the cluster can be represented with RLE codes by recording the number of occurrences. Furthermore, core-column, core-row, and region addresses are encoded as in the *remainder* part of Golomb codes [5]. The non-uniform distribution of the pixel values can be dealt with VLC, as in [3]. In particular, with Huffman coding [5] it is possible to reduce the redundancy approaching the entropy of the source. In this paper we follow a simplified scheme, devising a VLC prefix code that uses, for the two most probable values, the lowest number of bits, as in Huffman coding.

The presented algorithm can be operated either in column-by-column (*intra*-column) mode or in multiple column (*inter*-column) mode, increasing complexity. Indeed, in the latter, the compression module should have simultaneous access to all the FIFOs to detect if the considered cluster includes multiple adjacent columns.

The resulting code alphabet is summarized in Table I. In the first column, the codeword symbol, which is represented by an identifier and by a maximum of four optional parameters, has been reported. The word indentation is used to show the nesting relation between a symbol and its ancestor. The second column reports the binary codeword associated to the symbol, where characters correspond to the digits of the optional parameters (OP) in the symbol. The third column describes the symbol and reports the codeword bit size.

TABLE I
CODE ALPHABET FOR THE RLE-VLC ALGORITHM (ID = IDENTIFIER, OP = OPTIONAL PARAMETERS)

Symbol (ID, OP)	*Codeword bits*	*Description and size*
C,COL	$\underbrace{cc\ldots c}_{B_c \text{bits}}$	Column address B_c = no. of address bits
K,CORE,REG	$\underbrace{cc\ldots c}_{B_k \text{bits}}\underbrace{rr\ldots r}_{B_r \text{bits}}$	Cluster address ($B_k + B_r$ bits) B_k = no. of core address bits, B_r = no. of region address bits
F	1	Start of active region (1 bit)
T,A,B,C,D	$1\,\underbrace{aa\ldots a}_{B_p \text{bits}}\underbrace{bb\ldots b}_{B_p \text{bits}}$ $\underbrace{cc\ldots c}_{B_p \text{bits}}\underbrace{dd\ldots d}_{B_p \text{bits}}$	Active Region with *current* coding ($1 + 4B_p$ bits) B_p = no. of ToT bits
H	0	Active region with VLC, 1 bit
V1	0	1st most probable value, 1 bit
V2	11	2nd most probable value, 2 bits
V,VAL	$10\,\underbrace{vv\ldots v}_{B_p \text{bits}}$	All other values, $2 + B_p$ bits
R,RUN	$0\,\underbrace{rr\ldots r}_{B_{\text{run}} \text{bits}}$	Inactive regions, $1 + B_{\text{run}}$ bits
EOK	$0\ \underbrace{11\ldots1}_{B_{\text{run}} \text{bits}}$	End of cluster, $1 + B_{\text{run}}$ bits
EOC	$0\ \underbrace{00\ldots0}_{B_{\text{run}} \text{bits}}$	End of column, $1 + B_{\text{run}}$ bits

A. Spatial Compression of Active Regions

First, we deal with the uneven spatial distribution of the active regions. Figure 4 shows an example of an hit pattern that could be encoded as a single cluster. In the cluster, we can identify both active and inactive regions. Active regions naturally occur in clusters, and therefore it is more efficient to address the cluster instead of each single region. The adopted scan order proceeds from top to bottom and from left to right, following the same scheme used in the chip. As reported in Table I, B_k bits are associated with the address of the first core of the cluster, while B_r bits are used to address the first active region in the core. The inactive regions included in the cluster can be compressed with RLE prefix codes by encoding the number of occurrences L_R (with B_{run} bits). The run values $L_R = 0$ and $L_R = 2^{B_{\text{run}}} - 1$ are used to indicate the *end-of-cluster* (EOK) and the *end-of-column* (EOC) symbols, respectively. Therefore, the maximum run value can be $L_{R_{\max}} = 2^{B_{\text{run}}} - 2$.

Depending on the number of inactive regions between two clusters, it might be convenient to fuse two clusters into a single one: the full addressing is less convenient than the RL coding if

$$(1 + B_{\text{run}}) \lceil D_k / L_{R_{\max}} \rceil < B_k + B_r, \tag{1}$$

where D_k is the number of inactive regions between the two clusters.

The non-uniform statistical distribution of the ToT values is dealt with VLC, such as in [3]. In our approach, we adopted a simplified form of Huffman coding (that is a specific VLC code) by encoding the two most probable values with the smallest number of bits. The proposed algorithm dynamically

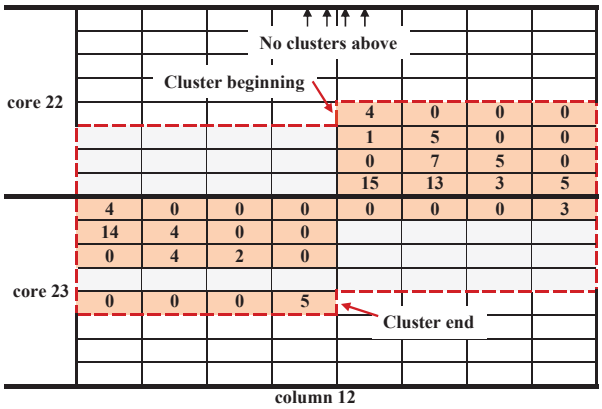

Fig. 4. Clustering for the intra-column mode of operation.

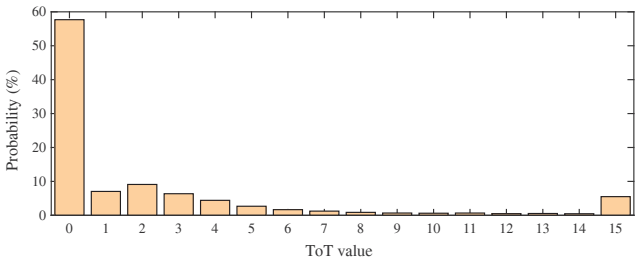

Fig. 5. Histogram of the pixel values in active regions.

chooses between the *current* coding and the VLC approach, depending on the values of the ToTs. It should be underlined that the VLC is advantageous over traditional coding when

$$N_1 (1 + B_p) + N_2 B_p > 2N_p, \tag{2}$$

where N_1 and N_2 represent the number of pixels with the first and second most probable ToT value in the specific active region, respectively, B_p is the number of bits used to encode the ToT value, and N_p is the number of pixels in a region.

B. Clustering and Bitstream Formation

The proposed spatial compression of active regions results in the formation of active region clusters, which can be represented by a binary bitstream. In Fig. 4, we consider an example obtained with $B_c = 6$, $B_k = 6$, $B_r = 4$, $B_p = 4$, $B_{\text{run}} = 3$. In the figure, the first cluster, occurring in column no. 12, has been enclosed within the dashed border. This cluster spans across cores no. 22 and 23 and across 16 regions, of which only 9 are active. The full symbol sequence produced by the proposed algorithm is reported in Table II. The first emitted symbol (C, 12) denotes the address of column 12, because it is the first cluster in this column. It is followed by the cluster symbol (K,22,9), specified in terms of core and region address. The first active region is encoded with VLC and the symbol H is emitted (since this is the first region in the cluster, it is active and the symbol F must be omitted). The first pixel in this region is encoded as a generic value (V,4), while the three following pixels have the most probable value (the ToT value 0), and thus they are encoded

TABLE II
SYMBOL SEQUENCE FOR THE EXAMPLE IN FIG. 4 ($B_c = 6$, $B_k = 6$, $B_r = 4$, $B_p = 4$, $B_{\mathrm{run}} = 3$)

Index	*Code symbols*	*Description*	*Index*	*Code symbols*	*Description*
0	C,12	First cluster of column no. 12	28	V,4	Pixel with ToT value 4
1	K,22,9	Cluster begins at core no. 22, region no. 9	29-31	V1 V1 V1	Three pixels with most frequent ToT value
2	H	Active region with VLC	32-33	F H	Active region with VLC
3	V,4	Pixel with ToT value 4	34-36	V1 V1 V1	Three pixels with most frequent ToT value
4-6	V1 V1 V1	Three pixels with most frequent ToT value	37	V,3	Pixel with ToT value 3
7	R,1	Run of one empty region	38-39	F H	Active region with VLC
8-9	F H	Active region with VLC	40	V,14	Pixel with ToT value 14
10	V,1	Pixel with ToT value 1	41	V,4	Pixel with ToT value 4
11	V,5	Pixel with ToT value 5	42-43	V1 V1	Two pixels with most frequent ToT value
12-13	V1 V1	Two pixels with most frequent ToT value	44	R,1	Run of one empty region
14	R,1	Run of one empty region	45-46	F H	Active region with VLC
15-16	F H	Active region with VLC	47	V1	Pixel with most frequent ToT value
17	V1	One pixel with most frequent ToT value	48	V,4	Pixel with ToT value 4
18	V,7	Pixel with ToT value 7	49	V2	Pixel with 2nd most frequent ToT value
19	V,5	Pixel with ToT value 5	50	V1	Pixel with most frequent ToT value
20	V1	One pixel with most frequent ToT value	51	R,3	Run of three empty regions
21	R,1	Run of one empty region	52-53	F H	Active region with VLC
22	F	Active region	54-56	V1 V1 V1	Three pixels with most frequent ToT value
23-26	T,15 13 3 5	Four pixels with ToTs 15, 13, 3, and 5	57	V,5	Pixel with ToT value 5
27	F H	Active region with VLC	58	EOK	End of the cluster

TABLE III
BITSTREAM OF SYMBOLS IN TABLE II

001100 – 0101101001 – 0 – 100100 – 0 – 0 – 0 – 0001 – 1 – 0 – 100001 – 100101 – 0 – 0 – 0001 – 1 – 0 – 0 – 100111 – 100101 – 0 – 0001 – 1 – 11111110100110101 – 1 – 0 – 000100 – 0 – 0 – 0 – 1 – 0 – 0 – 0 – 0 – 100011 – 1 – 0 – 101110 – 100100 – 0 – 0 – 0001 – 1 – 0 – 0 – 100100 – 11 – 0 – 0011 – 1 – 0 – 0 – 0 – 0 – 100101 – 0111

TABLE IV
RATE AND COMPRESSION RATIO OF THE TESTED ALGORITHMS

Algorithm	*R (bpp)*	*C*
Current	0.0338	1.0
RLE-VLC intra-column	0.0216	1.57
Huffman intra	0.0208	1.62
RLE-VLC inter-column	0.0212	1.59
GZIP	0.0286	1.18

by using three symbols for the most probable value (V1 V1 V1). A run of a single inactive region is then encoded (R,1), followed by one active region (F) in VLC mode (H). The four pixels are: one ToT value 1 (V,1), one ToT value 5 (V,5), and two most probable values (V1 V1). Then, another single inactive region is encoded (R,1). The active region symbol (F) has been used after this run, since another run could have been possible. Therefore, the maximum possible number of consecutive inactive regions in this example is 12, according to (1). Then, the encoding process continues until the cluster ends (EOK). The EOK should be replaced with the EOC symbol if the cluster is the last of the column.

The resulting bitstream is also reported in Table III, where each group of bits (separated by dashes) corresponds to an encoded symbol. The size of this bitstream is equal to 161 bits. Compared with the *current* encoding method (requiring in this case 288 bits), we reduce the number of bits by 44.1%, with a compression ratio of $C \approx 1.79$ (defined as the ratio between the sizes of the bitstreams generated by the *current* and by the considered algorithm).

C. Extension to the Tnter-column Case

Differently from the intra-column case, in the inter-column operation the compression circuit should take into account the position of active regions in all involved columns. Thus, a cluster is formed by exploring all the active adjacent regions, up to a pre-defined maximum cluster size. Then, the code alphabet is slightly modified, by adding a width parameter after the cluster symbol (K,CORE,REG). Therefore, the addition of inter-column compression increases the memory needs and introduces a larger complexity.

III. SIMULATION RESULTS

Performance is evaluated using MATLAB®, averaging the results over 342 triggers and neglecting the Aurora protocol encapsulation overhead, in terms of compression ratio C and in terms of average bit rate per pixel R (defined as the ratio between the size N_B of the bitstream generated by the considered algorithm and the number of pixels P of the chip, resulting in $R = N_B/P$ bpp).

In all cases, the compressed bitstream has also been decompressed, without any detected difference, decoding failure, and ambiguity. The performance of the proposed RLE-VLC algorithm has been compared to that of a full Huffman encoding implementation, where the probability of occurrence of the code symbols in Table I is estimated during a first encoding pass. Then, an optimized Huffman dictionary is

generated and used during a second encoding pass, to calculate the compression performance, without considering the size of the dictionary in the compression ratio calculation. Finally, we have considered the GZIP file encoding algorithm applied to the current bitstream.

In Table IV, the results show that the RLE-VLC intra-column method achieves a compression ratio $C = 1.57$, resulting in a bitstream size reduction of 36.2% with respect to the current bitstream size. On the other hand, full Huffman encoding in intra-column mode increases the compression ratio to $C = 1.62$, with a reduction of 38.4%. Instead, for the RLE-VLC inter-column method, the compression ratio is of $C = 1.59$, with a reduction of 37.2%. Finally, by considering the GZIP method, a smaller compression ratio has been obtained.

IV. Conclusion

In this paper, an improved algorithm for data compression, to handle the high data volume of the HL-LHC experiment upgrade, has been proposed for implementation in the next generation of the RD53A prototype chip, which is called RD53B. The algorithm exploits RLE and VLC compression techniques and its performance has been evaluated in MATLAB®. From the obtained results, we conclude that RLE-VLC in intra-column mode provides a good compression efficiency, whereas only moderate improvements are obtained switching to an inter-column mode or to a full Huffman implementation.

References

[1] RD53 Collaboration, "The RD53A integrated circuit," CERN, Tech. Rep. CERN-RD53-PUB-17-001, Nov. 2017.

[2] M. Garcia-Sciveres and X. Wang, "Data encoding efficiency in pixel detector readout with charge information," *NIMA*, vol. 815, pp. 18–22, 2016.

[3] S. Poulios, K. Androsov, M. Minuti, and F. Palla, "Lossless data compression for the HL-LHC silicon pixel detector readout," in *Proc. of MOCAST 2016*, Thessaloniki, Greece, May 2016, pp. 1–4.

[4] E. Conti, S. Marconi, J. Christiansen, P. Placidi, and T. Hemperek, "Simulation of digital pixel readout chip architectures with the RD53 SystemVerilog-UVM verification environment using Monte Carlo physics data," *Journal of Instrumentation*, vol. 11, no. 01, p. C01069, 2016.

[5] K. Sayood, *Introduction to data compression, 5th ed.* Cambridge, MA, USA: Morgan Kauffmann, 2018.