

# DYNAMIC RECONFIGURABLE HARDWARE FOR LOSSLESS COMPRESSION OF IMAGE, VIDEO AND GENERAL DATA CONTENT

*Jose Luis Nunez-Yanez, Xiaolin Chen, Nishan Canagarajah*

*Raffaele Vitulli*

Electronic Engineering Department  
Bristol University  
Bristol., BS8 1UB, UK  
e\_mail: {eejlny,eezxxc,eecnc}@bristol.ac.uk

European Space Agency (ESA)  
On-Board Payload Data Processing Section  
Keplerlaan 1, Noordwijk, 2200, The Netherlands  
e\_mail: raffaele.vitulli@esa.int

## ABSTRACT

This paper investigates an universal algorithm and hardware architecture for context-based statistical lossless compression of visual and data content using FPGA (Field Programmable Gate Array) devices which support partial and dynamic reconfiguration, enabling optimal modeling strategies for each data and compression type. Entropy coding of the modeling output is performed using a statically configured arithmetic coding engine. The current trend of network convergence where visual and data content are transmitted along the same physical channel suggests a technology capable of delivering optimal compression ratios and fast adaptation to the nature of the content will become increasingly important.

## I. INTRODUCTION

Lossless and lossy compression algorithms are routinely used to reduce the bandwidth and storage requirements of digital data. Lossy compression is well suited to data that is already a digital approximation of data that is analogue in nature such as visual and audio information. Lossy compression can achieve much higher compression ratios than lossless precisely because there is not a requirement to maintain all the information contained in the data source. Lossy compression has been widely adopted and global standards exist such as H.264 and MPEG4 for video and JPEG for still images. Traditionally, lossless compression has been used in those data types that do not admit any bits to be modified in the compression/decompression processes. Examples are general data types such as text, html code, database information, application data and program binaries where reversible compression is required since every bit contains critical information. Nevertheless, lossless compression of images and video is also an important topic of research. Medical images such as X-rays must be compressed without any loss since the implications could be catastrophic for the patient. Precious and hard-to-acquire images such as those obtained in space exploration and satellite surveillance also use lossless compression. Image archiving will benefit from a master copy stored

using a lossless approach from which copies of any desired quality could be obtained using lossy methods. Additionally applications such as data, video and image transmission in space require the performance to be achieved in an energy and silicon efficient platform. To achieve the demands set by these applications we propose a universal lossless compression hardware core that exploits dynamic reconfiguration to effectively combine predictive coding, motion estimation and context-based modeling hardware blocks depending on the data type.

## II. RELATED WORK

Current lossless compression for general data makes a distinction between dictionary-based and statistics-based algorithms. Dictionary-based compression has been traditionally more popular in software and hardware due to the inherent simplicity of these algorithms. Examples of dictionary-based compression in software are the popular WinZip or Arj algorithms commonly used for archiving and distributing large quantities of data. Also, the hardware devices available from leading companies such as HiFn Microelectronics LZS [1] use dictionary-based compression methods based on the original LZ [2] algorithms. Our research group's work has been extensively based on dictionary-based compression with the X-MatchPRO [3] algorithm targeting a compression ratio that halves the original input size while operating at very high speeds. This research area both in software and hardware is now mature with more than 25 years of work having been dedicated to fine tune the LZ method and few improvements can be expected. Context-based statistical compression has been limited to software and it is not very popular due to its high computing requirements. An example of a powerful statistical compressor is the PPMZ [4] software algorithm that requires more than 20,000 CPU cycles per byte in a general purpose microprocessor. Algorithmic research in lossless image compression has focussed in two main techniques; transform-based and predictive coding. Extensive experimentation seems to indicate that transform methods perform worse than predictive methods for lossless compression. Predictive coding is a technique where the value of the next input data

is predicted as a linear or nonlinear combination of previous inputs. It has been successfully used in lossless compression of visual content where it makes use of a priori knowledge of smoothness. Smoothness means that visual signals tend to follow a pattern of gentle variation that can successfully be exploited in the first processing stage to reduce the entropy of the data source. The output of the predictive coder can be modelled using a context dependent technique to further remove redundancy prior to being entropy coded using Huffman coding or arithmetic coding. Algorithms that employ this approach are the Sunset [5], Felics [5] and the LOCO [6] algorithm used in the JPEG LS. There are few hardware devices proposed for lossless image compression using predictive context-based arithmetic coding. A successful example targeted at the compression of black and white fax images is the IBM Q-Coder [7] device. This chip is based on a simple fixed high-order (7th) binary model. This simple model means that performance is not well suited for general alphabets. This device achieves a throughput of 64 Mbits/second when implemented in a CMOS 5S (0.35  $\mu\text{m}$ ) technology from IBM.

Hardware-based lossless video compression is a largely unexplored area. An example of a software lossless video codec is the Huffvuv algorithm heavily based on lossless JPEG operating on each of the frames of the sequence. Popular lossy video codecs such as H.264 also offer a lossless mode [8].

### III. DYNAMICALLY RECONFIGURABLE MODELING STAGE

The proposed compression system uses a dynamically reconfigurable modeling stage followed by statically configured probability estimation and arithmetic coding stages as illustrated in Fig.1. Dynamic modeling is specialized to each data type and uses a combination of context modeling, predictive coding and motion estimation depending on the data type being processed: data, image or video.

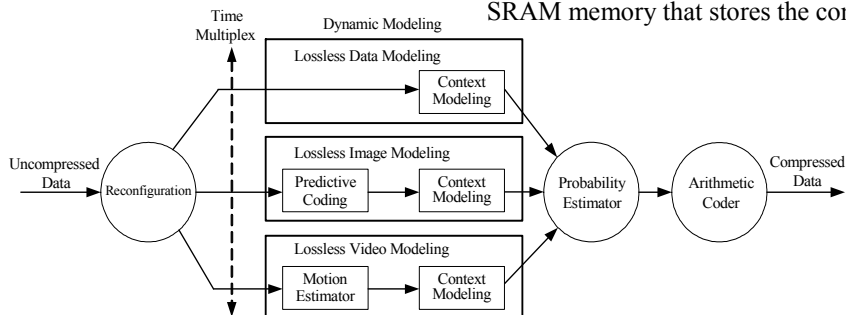


Figure 1. System overview

#### A. Lossless Data Modeling overview

During context modeling for data a finite number of symbols (model order) that preceded the current symbol in a single dimension and constitute its context are searched in a context tree built dynamically as more data is seen. Fig. 2 shows a simplified diagram of the context modeller. The context FIFO stores the symbols that preceded the current symbol and form its context. The FIFO width is 1 byte to match the width of the symbol while its length is configurable and depends on the maximum model order.

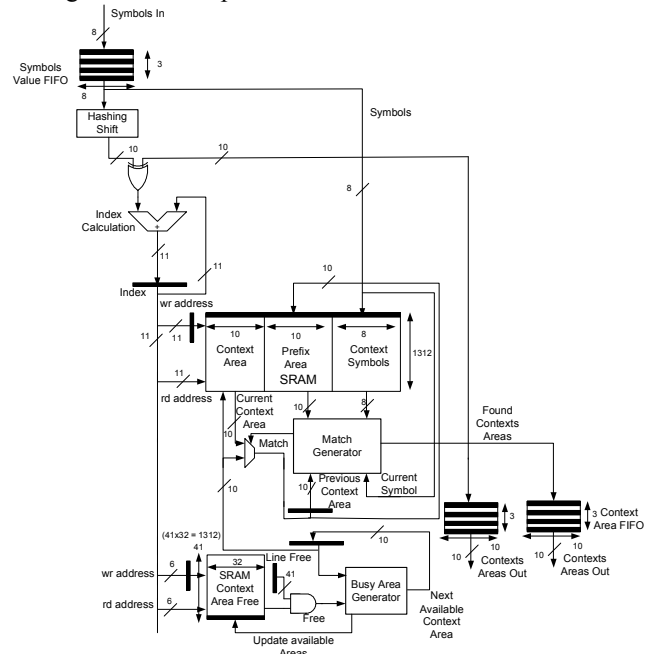


Figure 2. General data context modeling architecture

The hardware implementation of the context modeller is based on a hashing tree that enables fast search operations with low complexity. The tree is stored in standard SRAM memory and maintains its logical structure using a pointer mechanism. The hashing shift and the XOR gate in Fig. 2 are used to generate an index to be used to address the SRAM memory that stores the context tree.

The tree memory is divided into three sections. Section 1 stores the context area memory address where the probability data for that particular tree node can be found. The other two sections implement the pointer mechanism that maintains the logical structure of the tree. Section 2 stores the context area index of the tree node parent of the current node in the tree structure. Section 3 stores the context symbol stored at the current tree node. The SRAM area free memory and the busy area generator shown in Fig. 2 enable a single-cycle reset state without having to reset the table memory with a multi-cycle table walk operation. A table walk would have had a very negative effect on throughput when dealing with small blocks since the number of cycles needed to reset the table could typically be larger than the number of cycles needed to compress the block. A single valid register, named line free in Fig. 2, is reset after processing each block and this automatically invalidates all the locations in the table memory. This register has a similar function to the register holding the valid bits in a direct-mapped cache. Each of the register bits is shared by several table locations and in order to distinguish which context tree nodes are busy and which context tree nodes are free the area free memory contains 1 bit per context tree node signaling a free or busy node. If the valid register bit is set to zero all the tree nodes associated with that valid register bit are considered invalid. The found context areas are stored in two equivalent buffers. When the first buffer is being filled with context areas by the context modeler the second buffer is being emptied by the probability estimator. Once both stages have completed their operation the buffers functionality is reversed and the process restarted. This double buffering mechanism increases the throughput of the system avoiding idle stages.

## B. Lossless Image Modeling overview

Lossless image modeling is often divided into two phases: prediction and context modeling. Prediction tries to make use of the notion of data smoothness characteristic of images and not present in general data. Heuristic prediction that requires only a few fixed parameters is preferred to other complex universal models. In this work, the predictor of the image modeling is inspired by the predictor GAP (Gradient-Adjusted Prediction) from CALIC [9], a software-based algorithm. However, our method is substantially simplified and amenable to hardware implementation. A two dimensional grid formed by 7 neighboring symbols of the current symbol constitute the context of the current symbol. In the prediction phase, we estimate the local edge feature by calculating the vertical and horizontal intensity gradients  $dv$ ,  $dh$ , using the differences between context symbols vertically and horizontally. The predicted symbol value is the linear combination of its contexts, according to the gradient

direction and magnitude. The predictor is designed to be suitable for hardware, involving only addition/subtraction and shifting. Thus we obtain the prediction error  $e$ , the difference between the original and the predicted symbol value, for context modeling. In the context modeling phase, context selection is essential to reduce memory usage. We use 6 context symbols to compare with the predicted value to obtain a texture pattern  $t$ , representing the local texture feature. Also, to indicate the activity of errors in a context, a coding context is generated with the local gradients  $dv$ ,  $dh$  and a previous prediction error  $W$ . The coding context is quantized into 8 levels to produce a coding context index  $QE$ . Combining the texture pattern and coding context, a set of 512 compound contexts are formed with 6 bits texture pattern  $t$  and 3 bits coding context index  $QE$ . These contexts are also used to generate an error feedback to adjust the bias of prediction. Fig. 3 presents the hardware architecture of the image modeling module, where the prediction stage is illustrated in the solid-line box and the context modeling stage is the rest.

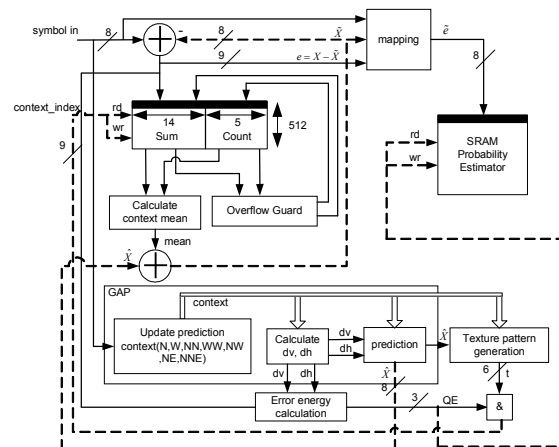


Figure 3. Image modeling architecture

The implementation is achieved with two pipelines running in parallel. Line 1, indicated by solid line in Fig.3, operates on the current symbol and yields the prediction error  $e$  and coding context index  $QE$  for the probability estimator. Line 2, indicated by dash line, calculates the prediction value and context index for next symbol. So the Line 1 operation for the next symbol can utilize the result of Line 2 from current symbol when the two lines work in parallel. Details on how to manage the context memory and the error feedback technique, which calculates the mean of errors in a context are given below. For the compression process, we need to store 3 lines of image pixel values in memory as context and use 3 pointers to indicate symbol locations in each line. At the end of processing each line, the 3 pointers to each line have to be rotated in a certain order so that the oldest line will be discarded and the newly formed line will be saved.



stream. The whole process is numerically efficient and using 9 coding events instead of 1 coding event per input symbol produces no significant redundancy. Fig. 4 illustrates the architecture of the processing element that implements the binary tree node assuming a context population of 1024. The total value memory contains the total frequency count for a particular context while the probability storage memory contains all the probability data associated with each of the nodes in the tree.

### B. Arithmetic Coding

The final stage of the coding process is arithmetic coding. The arithmetic coder is based on a software algorithm known as the Z-coder and developed by AT&T labs as a generalization of the Golomb/Rice coder for lossless coding of bilevel images. Our work has focused on maintaining the simplicity of the Z-coding algorithm while increasing its suitability for hardware implementation. The resulting MZ-coder balances the complexity of coding the MPS and LPS symbols, simplifies the precision of the arithmetic and handles special hardware borrow conditions while maintaining coding efficiency and achieving high performance. Fig.5 shows the internal organization of the multiplication-free arithmetic coding module. A total of 6 pipeline stages are identified to improve the clock ratio of the design. The lack of a renormalization loop in the MZ algorithm means that one decision bit is processed per clock cycle. The functionality of each of the pipeline stages is explained in detailed in [10].

## V. PERFORMANCE COMPARISON

This section analyses the performance of the core in terms of compression ratio and throughput and compares it with other compression algorithms for data and images implemented in both hardware and software.

Table 1 compares the compression efficiency of the algorithm configured in data mode with two well-known algorithms using the Canterbury corpus as the data set. We have selected the popular open source Lempel-Ziv implementation known as GZIP and equivalent to other commercial algorithms such as PKZIP and WinZIP as a fast and efficient dictionary-based algorithm. LZS is targeted to hardware as described in the related work section. The table measures compression in bits per byte and shows how the proposed method outperforms both GZIP and LZS.

We compared the performance of the proposed method configured in image mode in Table 2, using a set of grey-scale standard test images of size 512\*512 pixels. JPEG-LS (LOCO-I) and SLP (Switched Linear Prediction) are low complexity compression schemes using Golomb-Rice coder. Clearly, the proposed scheme outperforms JPEG-LS and SLP in terms of compression ratio although the improvement is smaller than for the general data case.

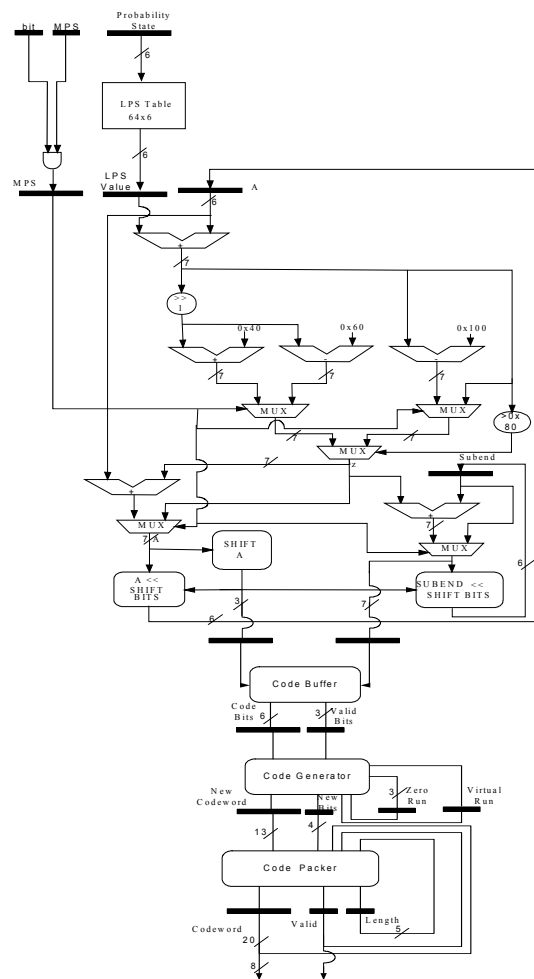


Figure 5. Arithmetic coding architecture

Table 1. General data compression performance

File	GZIP	LZS	Proposed (data mode)
alic	2.86	4.19	2.43
asyo	3.05	4.29	2.57
cphp	2.54	3.64	2.48
fiel	2.20	2.87	2.16
gram	2.62	3.09	2.51
kenn	1.60	2.31	1.37
lcet	2.71	4.14	2.25
plra	3.24	4.69	2.49
ptt5	0.87	1.26	0.89
summ	2.70	3.54	3.10
xarg	3.29	3.89	3.14
average	2.52	3.45	2.30

Table 2. Image data compression performance

Image	JPEG-LS	SLP(M0)	Proposed (image mode)
barb	4.86	4.79	4.68
boat	4.25	4.28	4.18
goldhill	4.71	4.74	4.65
lena	4.24	4.17	4.14
mandrill	6.04	5.99	5.93
peppers	4.49	4.49	4.39
zelda	4.01	3.97	3.90
average	4.66	4.63	4.55

Fig. 6 shows the constrained layout of the implementation in a SX35 Virtex-4. This device supports partial dynamic reconfiguration and offers enough logic resources to implement several cores working in parallel to deliver better performance. The design achieves a clock frequency of 100 MHz, which produces approximately a throughput of 100 Mbits/sec. The low complexity means that a multi-core solution could be used to scale up performance.

Table 3. Resource utilization summary

	Dynamic modeling		Static Coding	
	Image	Data	Probability Estimator	Arithmetic Coder
No. of Slices	526	213	297	1123
No. of Slice Flip-flop	217	145	124	283
No. of 4 input LUT	902	365	561	2131
No. of RAM16s	6	2	9	0

The area in the red box corresponds to the reconfigurable area that implements the dynamic modeling technique. The figure shows the device configured in data mode. Since the resources used for image modeling are higher than for data modeling the number of logic cells used in the red box is higher in this case.

## VI. FUTURE WORK AND CONCLUSIONS

We are currently working in adding lossless video compression support designing an efficient pixel-oriented vector-less motion estimation engine. We expect to be able to process the residue resulting from motion estimation using a context modeling technique similar to the one used in the image mode. We would also like to investigate the configuration of different alphabet sizes extending the current byte-based alphabets to multiple-bit alphabets for lossless compression of scientific data obtained from high-resolution analogue-to-digital converters. The analysis of the compression performance has shown this method to be very competitive while the partial reconfiguration features should enable a silicon and energy efficient platform. Executables and information for this core named Byacom-2 will become available at [www.byacom.co.uk](http://www.byacom.co.uk) as the project

progresses. We would like to acknowledge the support of EPSRC under grant number EP/D011639/1 for making this research possible.

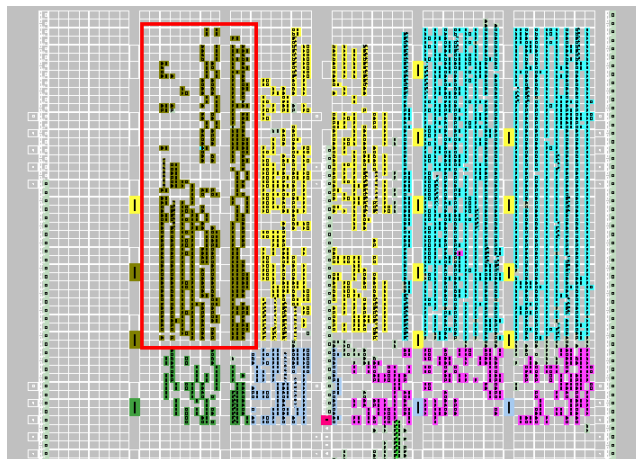


Figure 6. Compression core layout

- [1] '9600 Data Compression Processor', Data Sheet, Hi/fn Inc, 750 University Avenue, Los Gatos, CA, 1999.
- [2] J.Ziv, A.Lempel, 'A Universal Algorithm for Sequential Data Compression' IEEE Trans. Inf. Theory, Vol. IT-23, pp. 337-343, 1977.
- [3] J.L. Núñez, S. Jones, 'Gbit/Second Lossless Data Compression Hardware', IEEE Transactions in VLSI Systems (TVLSI), Vol. 11, No. 3, pp. 499-510, June, 2003
- [4] C. Bloom, 'Solving the Problems of Context Modelling', <http://www.cbloom.com/papers/index.html>, 1998.
- [5] X. Wu, 'An algorithmic study in lossless image compression', Proceedings of the 1996 Data Compression Conference, Snowbird, Utah, pp. 150-159, April 1996.
- [6] M. Weinberger, G. Seroussi, and G. Shapiro, 'The LOCO-I lossless image compression algorithm: Principles and standardization into JPEG-LS', IEEE Trans. on Image Proc., vol. 9, pp. 1309--1324, Aug. 2000.
- [7] M. J. Slattery, J. L. Mitchell, 'The Qx-Coder', IBM Journal of Research and Development, Vol. 42, No. 6, pp. 767-784, 1998.
- [8] Sullivan, T. Wiegand, "Video Compression - From Concepts to the H.264/AVC Standard", Proc. the IEEE, Special Issue on Advances in Video Coding and Delivery, Vol. 93, No. 1, pp. 18-31, January 2005.
- [9] X. Wu and N. Memon, "Context-based, adaptive, lossless image coding," IEEE Trans. Commun., vol. 45, no. 4, pp. 437-444, Apr. 1997.
- [10] J. L. Nunez-Yanez and V. A. Chouliaras, "A configurable statistical lossless compression core based on variable order Markov modelling and arithmetic coding," IEEE Trans. Comput., vol. 54, no. 11, pp. 1345-1359, Nov. 2005.