

# Design and Implementation of an Arithmetic Codec for Wavelet based Image Compression

## Abstract

The implementation of an arithmetic encoder/decoder as part of a Wavelet based video codec for a mobile multimedia communicator prototype is presented in this paper. Based on an adaptive model for representing the data to compress, the best possible compression code in the sense of minimum average code length is very nearly achieved. A low precision implementation has been developed minimizing the well known problems of speed and complexity of the arithmetic codecs.

## 1. Introduction

In any communications system, the idea of compressing the information to be transmitted is very attractive, due to the limitations of the available bandwidth. When the amount of data to transfer is too large, the data compression is not only an attractive idea but also a necessity. This is the case of video and image transmission.

For this reason, the functionality of any system aimed to send and receive images can be divided into three main stages: image capture, image processing and compression of the data supplied by the image processing stage.

The strategy implemented on the mobile multimedia prototype being developed consists on carry out the Wavelet transform to obtain the coefficients that represent the image in the frequency domain. As most of the energy of any image is concentrated at low frequencies, most of all obtained coefficients can be ignored because they do not provide significative information [1]. The selection of the coefficients that are or not important to rebuild the image through the inverse Wavelet transform, is accomplished by a zerotree encoder, that gives as result a significance tree composed by the zerotree symbols and, only for the significative symbols, the associated Wavelet coefficients. If these symbols and coefficients are passed to an entropy encoder, the final result is a version of the initial image that takes much less storage space than the original one[1].

A remarkable aspect about this entropy encoder is that it must carry out a lossless compression, since it

is essential that after the decompression process the same original symbols and coefficients are obtained again at the receiver, in order to get the same significance tree, complete the inverse Wavelet transform, and rebuild the image successfully.

Lossless data compressors are based on the fact that in the information to compress there is some redundancy since there are events (symbols and coefficients) that take place more frequently than others, i. e., they have a greater occurrence probability [2]. A data compressor reduces the size of data provided by an information source, deciding which data occur more frequently and encoding spending fewer bits than the bits used with the less probable symbols or coefficients.

## 2. Arithmetic Coding

There are a lot of strategies and algorithms for lossless data compression. Some of them are very extended, as it is the case of Huffman method [2]. However, as it was theoretically and empirically proved during the 80's by Witten, Neal and Cleary [3], arithmetic coding is superior in most aspects to the Huffman algorithm. After that, P. G. Howard and J. S. Vitter [4] studied different practical implementations for the arithmetic coding during the 90's and concluded the convenience of choosing this kind of entropy encoder instead of any other in a multisymbol alphabet scenario as it is the case of Wavelet coefficients.

The most important advantage of arithmetic coding is its flexibility. Due to the clear separation between the model for representing data and the information encoding, the arithmetic encoder can easily be adapted to any kind of application by only changing the model [3]. In fact, an adaptive model can be used in order to adjust the probabilities of the data as they occur. The other important advantage of the arithmetic coding is its optimality. Arithmetic coding is optimal in theory and close to optimal in practice, regarding encoding using minimal average code length. On the other hand, the main disadvantage of arithmetic coding is its complexity and that it tends to be slow. The full precision form of arithmetic coding requires at least one multiplication per symbol/coefficient codification and in some implementations up to two

multiplications and two divisions per symbol/coefficient codification. The main contribution of this paper is, however, the use of low precision arithmetic, replacing multiplication and division operations with fast shift/add operations, minimizing the complexity of the algorithm and speeding up the processes of encoding and decoding.

### 3. Arithmetic codec architecture description

In arithmetic coding a message is represented by a subinterval of real numbers between 0 and 1. As the message becomes longer, the subinterval needed to represent it becomes smaller, and the number of bits needed to specify that subinterval grows. Successive symbols of the message reduce the size of the subinterval in accordance with the symbol probabilities generated by the model. The more likely symbols reduce the range by less than the unlikely symbols, and hence add fewer bits to the message. Before any data is transmitted, the range for the message is the entire interval [0, 1). As each symbol is processed, the range is narrowed to that portion allocated to the symbol. Graphical and numerical examples are shown in [3] and [4].

There are many implementations of these algorithms for two level signals [5][6], but this is not the case in multilevel signals, whose handling is more complex. This is partly due to the fact that the memory requirement is much larger, as it is necessary to store the probabilities of each of the possible symbols. Besides, the few arithmetic coding implementations for multilevel signals are generic and use to be only for research purposes [7][8]. The arithmetic coding implementation presented in this paper is designed and adapted for a specific real time application as it is the compression and transmission of images, instead of compression of a text or image file previously known.

The main aim of the arithmetic encoder described in this paper is to reduce redundant information of coefficients from the Wavelet transform and symbols from the zerotree encoder. In order to obtain the maximum reduction of the redundancy, the strategy to follow is the implementation of an arithmetic encoder based on an adaptive model for the probability distributions of the coefficients and symbols to encode. The implemented architecture is based on the low-precision arithmetic coding implementation of R. M. Neal [9], which is based on [3], but that provides several advantages with respect to the complexity of the algorithm and the speed of its execution. Some changes in the Neal implementation with respect to the Witten's version that had become a *de facto* standard, are that the arithmetic coding operations have been fiddled so that the division operation involved can be performed to very low precision. The model is now

required to produce "partially normalized" frequencies, in which the counter of the total occurred symbols is always more than half the maximum allowed for that counter. Besides, the model must now arrange for the most probable symbol to have index 1. However, there are several differences between Neal's implementation and the version described in this paper. The arithmetic codec presented in this paper is a hardware implementation based on the Neal's software description of the algorithm. Firstly, as it is a hardware implementation, the different functions and procedures of the software version were deeply studied to parallelize the tasks that could be done concurrently. Another important question is that Neal's implementation is a generic realization for text compression with the only purpose of showing the efficiency of the low-precision arithmetic coding in terms of compression ratio. However in this case, arithmetic coding algorithm has been adapted for a specific application in which the data to compress are zerotree symbols and Wavelet coefficients that come from a processed image. It means that in the implementation presented in this paper there are two different alphabets instead of one, and that the control and synchronisation strategy of the two different operation modes is included.

Input data for the encoder are coefficients or symbols depending on the working mode in which the encoder is. The coefficients' alphabet is composed by binary numbers that can be coded with eight bits and the symbols alphabet consists on 01, 10 and 11 symbols.

The arithmetic encoder encodes messages of N symbols or M coefficients. In the case of symbol codification, the special end-of-message 00 symbol delimits the string of symbols to be encoded. This end-of-message has the only function of delimitation and it is not included in the group of possible symbols from the zerotree encoder. In the case of

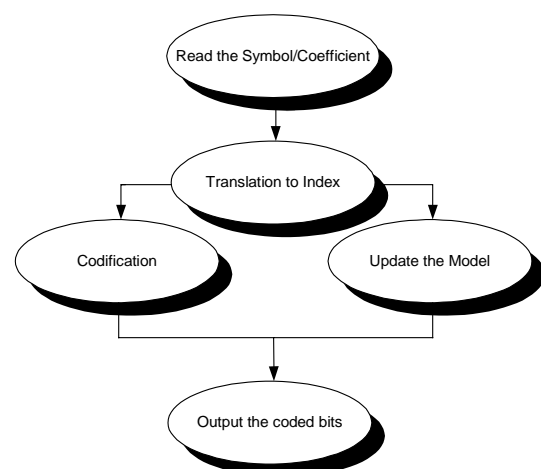


Fig. 1. Steps of the codification process

coefficients codification, as the number of coefficients in the string to be encoded can be previously known (only the symbols with their most significant bit equal to '1' have an associated Wavelet coefficient), counting the number of coefficients already encoded, the string can be easily delimited. In Fig.1 the different steps of the codification process are shown. Firstly, the symbol or coefficient depending on the operation mode is read. After that, the associated index is obtained. This is the way of knowing if the current symbol or coefficient is very probable or not, since one of the characteristics of the model is that the most probable symbols or coefficients are the smaller associated index they have. When the index is read, the codification and updating of the model can be done in parallel to save time. Finally, after the codification process, an output buffer is in charge of formatting the output coded stream, sending out the coded data asynchronously when an 8-bit output buffer is full.

The operation of the arithmetic decoder is just the opposite to the arithmetic encoder. The steps of the decodification process is shown in Fig. 2. The coded stream is received serially through an input buffer until a 32 bit word is completed, which is sent to the Decoder subsystem to start the decodification. When the symbol or coefficient is decoded, the model can be updated. As the symbols and coefficients are decoded and reconstructed, they are passed out of the arithmetic decoder in order to reconstruct the significance tree and the image processing blocks of the receiver prototype can recover the transmitted image trough the inverse Wavelet transform.

In the next section the functionality of the arithmetic codec subsystems is described.

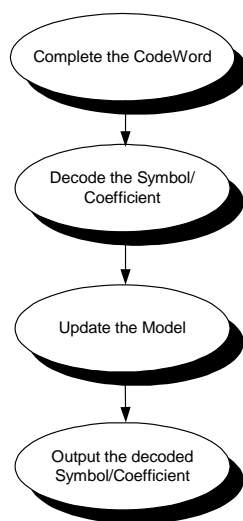
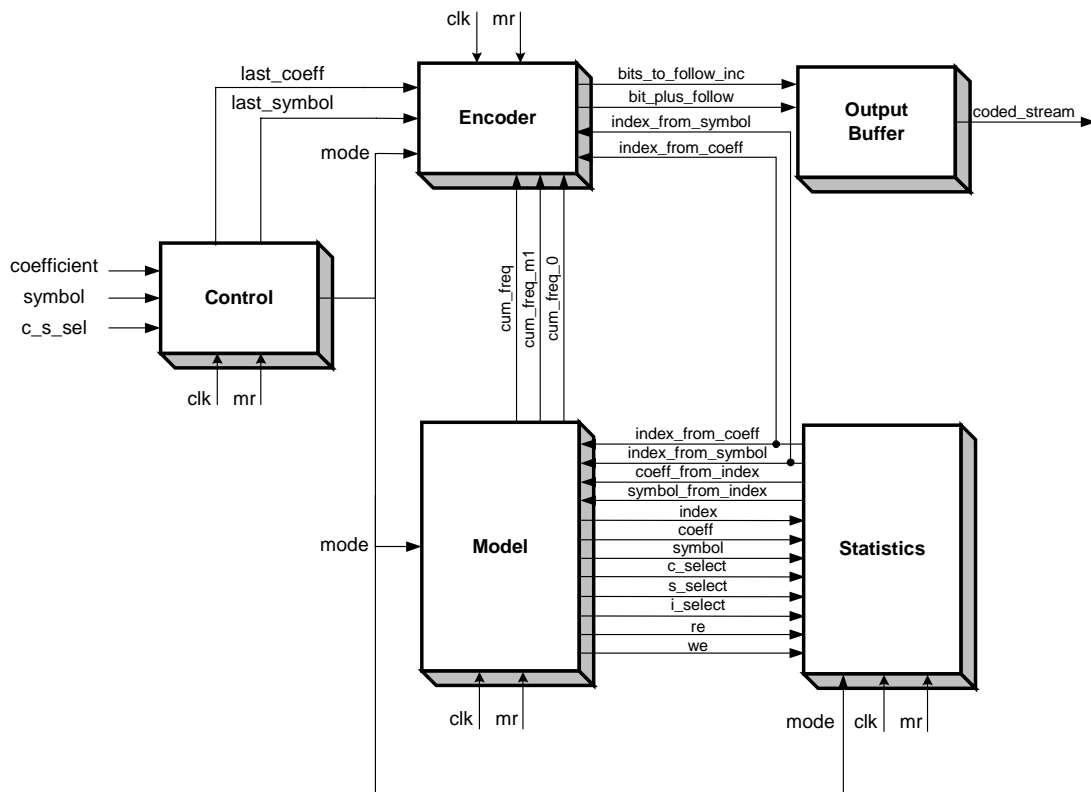


Fig. 2. Steps of the decodification process

#### 4. Arithmetic codec subsystems description

The arithmetic encoder and decoder are divided into several subsystems according the different functions involved in the process of codification and decodification of symbols and coefficients. The subsystems composing the architectures of the arithmetic encoder and decoder are shown in Fig. 3 and Fig. 4. These subsystems and the tasks carried out by each one of them are briefly described below:

- a) **Control subsystem:** This subsystem is in charge of all the operations related to the control of the different blocks. It sets the encoder to its initial state for beginning the codification of messages and communicates the current working mode (symbols or coefficients) to the necessary blocks. Control subsystem also counts the number of zerotree symbols that carry a wavelet coefficient in order to know when to stop coding coefficients and switch to the symbols mode again. Therefore the most important task of the Control subsystem is to identify when to switch between coefficient or symbol mode. An interface with the Master Control of the full mobile communicator architecture is also included.
- b) **Statistics subsystem:** The algorithm implemented for arithmetic encoding is based on having the different symbols sorted by their frequencies of appearance. For this reason, it is necessary to carry out a translation between the symbol to be encoded and its corresponding index, and viceversa by means of translation tables. These translation tables are allocated in the Statistics subsystem. As the values in the translation tables need to be modified too, some logic for controlling the writing and reading operations is necessary, as for example the write enable, read enable, and selection signals for symbols and coefficients.
- c) **Model subsystem:** As it was described before, there is a clear separation between the model for representing data and the information encoding. In this subsystem all the matters regarding the model used for the arithmetic encoder/decoder are included. Some of the parameters involved in the modeling of the information are the cumulative frequencies of the different symbols or coefficients and the total number of symbols or coefficients appeared. The updating of the model each time a symbol or a coefficient is encoded consists on incrementing these frequency counters. For this reason, another function of the Model subsystem is to control the overflow of the frequency counters.



**Fig. 3.** Arithmetic encoder scheme

- d) **Encoder subsystem:** This subsystem computes the coded value that corresponds to the message (sequence of symbols/coefficients) at the input of the arithmetic encoder. The bits obtained after the codification process will compose the final codeword, and are function of the different symbols or coefficients that have occurred and their cumulative frequencies at each moment, since this arithmetic encoder uses an adaptive model and the frequencies and probabilities are continuously changing. The subsystem has two main tasks to carry out. The first one is to narrow the subinterval that represents the sequence of symbols or coefficients that have occurred, according to the cumulative frequencies supplied by the Model subsystem. The second task is to output the already coded bits to an output buffer.
- e) **Decoder subsystem:** This module performs all the arithmetic operations needed by the decoding process, using the cumulative frequencies of the symbols and coefficients supplied by the Model subsystem.

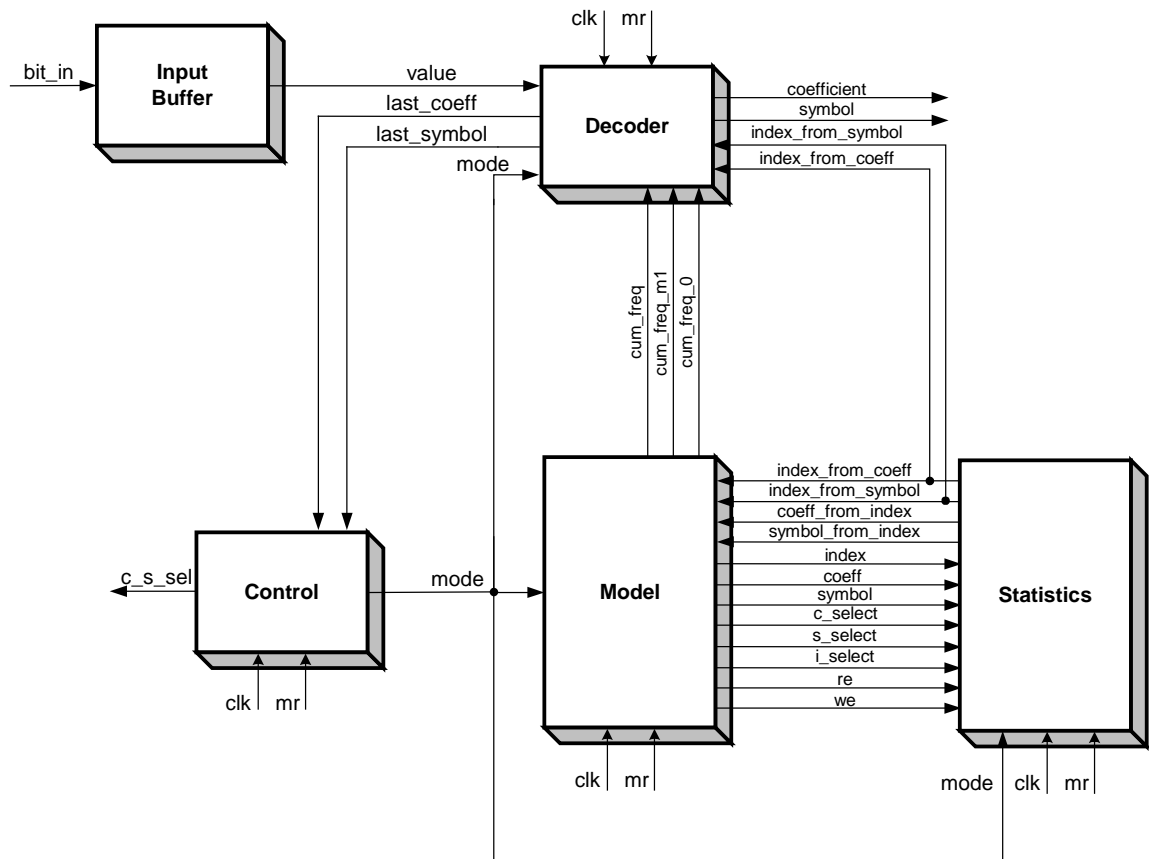
As it is shown in Fig. 3 and Fig. 4, with all these subsystems and some buffers (one output buffer at the encoder and one input buffer at the decoder) the full architecture of the arithmetic codec is complete. The Statistics and Model subsystems have an

identical implementation at the encoder and decoder as the functions to be carried out by these subsystems in both sides are the same. However, they have to be duplicated since the encoder and decoder of the same chip will be encoding and decoding different streams and therefore, the encoding and decoding contexts will be different. Based on low precision arithmetic, the architecture just described replace multiplication and division operations with fast shift/add operations, minimizing the complexity of the algorithm and speeding up the processes of encoding and decoding, that are the well known problems of arithmetic coding.

## 5. Implementation

The depicted architecture was described using the hardware description language *Verilog*, obtaining a fully synthesizable code. Different preliminary versions were developed until the final design was reached. Firstly a stand-alone coefficients codec was implemented and exhaustively tested and subsequently, the same steps were followed with the symbols codec. Finally, the functionalities of the independent codecs were joined adding the necessary logic for the control and synchronization of both operation modes.

The different blocks composing the arithmetic encoder and decoder are basically the ones described



**Fig. 4.** Arithmetic decoder scheme

in the preceding section. Both architectures can be divided into three main functionalities: the Encoder/Decoder core, the Model/Statistics and the Control. The Encoder/Decoder submodules implement the arithmetic necessary for the processes of encoding and decoding symbols and coefficients, depending on the operation mode in which the system is at each moment, and being based on the probabilities associated to the corresponding symbol or coefficient. These probabilities are supplied by the Model submodule, that carries out the control of occurrence of all the symbol and coefficients and communicates with the Statistics submodule, in order to make the translations between the symbols or coefficients and their corresponding indexes. Since the architecture is based on an adaptive model for representing the information, another important task of the model is its correct updating each time a symbol or coefficient is encoded or decoded. At last, the Control submodule is in charge of setting the operation mode of the different submodules that compose the architecture, taking into account that the encoding/decoding process of symbols ends when the end-of-message symbol, 00, occurs and that the number of coefficients to encode/decode is just the number of significative symbols encoded/decoded previously.

Regarding the test strategy, an exhaustive three levels verification scheme was followed to ensure the correct codification of the system at each of its stages. Based on the R. M. Neal low-precision implementation, a software description of the arithmetic codec was developed in C, in order to generate the different test patterns that were introduced to the *Verilog* description of the architecture. Firstly the different submodules were independently tested. After that, at the second level of the verification scheme, the different submodules composing the encoder and the decoder were connected and verified separately. Finally, when the correct functioning of the encoder and decoder was confirmed independently, they were connected in cascade in a way in which the *Verilog* decoder decoded correctly the coded stream provided by the *Verilog* encoder. This three level verification scheme was followed firstly with the coefficients arithmetic codec prototype and secondly with the symbols prototype. When the Control submodule was finished the full functionality of the symbols/coefficients arithmetic codec was tested, firstly focusing on the encoder and decoder separately and later connecting them in cascade as it was done with the prototypes. All the simulations were done with two different kinds of data: random

test pattern generated from the C high level implementation of the architecture, and zerotree symbols and Wavelet coefficients obtained from real images processed in *Matlab*. These last simulations, based on common images, were made recreating the scenario in which the arithmetic codec will be operating after its integration on the full mobile multimedia architecture.

Synthesis and implementations results, as well as power dissipation and area concerning this architecture will be included in the final version of the paper.

## 6. Conclusions

In this paper a specific application hardware implementation of an arithmetic codec has been described. Although arithmetic coding is a very flexible algorithm that can be used in any kind of application just choosing the appropriate model for representing the source to compress, the majority of the implementations are always limited to text applications. Besides, there are many implementations of this algorithm for two level signals, but this is not the case in multilevel signals, whose handling is more complex. However in this case the specific application of a multimedia communicator has benefited from the advantages of arithmetic coding, adapting the algorithm to a specific real time application as it is the compression and transmission of images. Based on a Wavelet image compression strategy, the implementation described in this paper is different to others image compression architectures based on arithmetic coding, because these last ones use to be binary implementations that only compress the sequence of bits that compose the image, and the implementation presented here is designed and adapted to a multisymbol alphabet scenario as it is the case of Wavelet coefficients and zerotree symbols. In fact, it is as if two independent arithmetic codecs for symbols and coefficients would be developed but adding the control and synchronisation for switching between the two operation modes.

Besides, the *a priori* disadvantages of arithmetic

coding, as it is the complexity of the algorithm that could have affected negatively to the power consumption, and the speed of the algorithm have been minimised choosing a low-precision implementation that avoids to perform multiplication and division operations.

The arithmetic encoder/decoder described in this paper will be part of a Wavelet based video codec chip for mobile multimedia communications purposes, that will be shortly in the market as a commercial product.

## References

- [1] José Oliver Gil , Manuel Pérez Malumbres “Compresión de imagen y vídeo: fundamentos teóricos y aspectos prácticos” Valencia - Universidad Politécnica, 2001
- [2] D.A. Huffman, “A method for the construction of minimum-redundancy codes”, Proc Institute of Electrical and Radio Engineers, 40, pp. 1098-1101, Sep. 1952
- [3] I.H. Witten, R. M. Neal, J. G. Cleary, “Arithmetic Coding for Data Compression”, Communications, ACM, 30, pp. 520-540, Jun. 1987
- [4] P. G. Howard, J. S. Vitter “Practical Implementations of Arithmetic Coding”, Image and Text Compression, J. A. Storer, Ed. Norwell, MA: Kluwer Academic Publishers, pp. 85-112, 1992
- [5] R. B. Arps, T. K. Truong, D. J. Lu, R. C. Pasco, T. D. Friedman, “A multi-purpose VLSI chip for adaptive data compression of bilevel images”, IBM J. Res. Develop., vol. 32, no. 6, pp. 775-795, 1988
- [6] W. B. Pennebaker, J. L. Mitchell, G. G. Langdon Jr, R. B. Arps, “An overview of the basic principles of the Q-Coder adaptive binary arithmetic coder”, IBM J. Res. Develop., vol. 32, no. 6, 1988
- [7] R. R. Osorio, J. D. Bruguera “New arithmetic Coder/Decoder architectures based on pipelining”, IEEE International conference on application specific systems, architectures and processors, pp. 106-115, 1997
- [8] J. Jiang, S. Jones, “Parallel design of arithmetic coding”, IEE Proc-Comput. Digit. Tech., vol. 141, no. 6, pp. 327-333, 1994
- [9] R. M. Neal, “Low precision arithmetic coding implementation”  
[ftp://ftp.cs.toronto.edu/pub/radford/lowp\\_ac.shar](ftp://ftp.cs.toronto.edu/pub/radford/lowp_ac.shar)