Reduction of Frames Storage Size in AFDX Reception End-System using a Lossless Compression Algorithm

Yohan Baga*†, Fakhreddine Ghaffari*, David Declercq*, Etienne Zante†, Michael Nahmiyace†
*ETIS, UMR 8051 / ENSEA, University of Cergy-Pontoise, CNRS, F-95000, Cergy-Pontoise, France
Email: (yohan.baga, fakhreddine.ghaffari, david.declercq)@ensea.fr
†ZODIAC AEROSPACE, Zodiac Aero Electric, Zodiac Cockpit and Lighting Systems,
7 rue des longs quartiers, 93100 Montreuil, France
Email: (etienne.zante, michael.nahmiyace)@zodiacaerospace.com

***Abstract*—The growth of bandwidth needs and reliability requirements has determined Avionics Full-Duplex Switched Ethernet (AFDX) networks as the new generation of on-board communication mediums. AFDX belongs to the deterministic, real-time and Ethernet-based network family. The AFDX terminals are called End-Systems (ES). The frames arriving at an ES have to be stored in a reception buffer to avoid frames losses or corruptions due to slowdowns in the ES layers. Little attention is carried to the issue of buffer dimensioning which is generally set to a very large size. However, a too large buffer size leads to costs in terms of memory resources and energy.**

In this paper, we propose to reduce the reception buffer size by using an LZW-based compression algorithm implemented in hardware. To do that, we interpret frames as sequences of hexadecimal source symbols, and we use a set of 4 parallel dictionaries to encode sequences of source symbols in fix-length words. We realize compression gain measured on sets of frames comprising several millions of symbols, and we obtain until 22% of memory gain when the dictionaries sizes are optimally dimensioned.

***Keywords* –** AFDX, Deterministic Network, Back-to-back Frames, Worst Frame Backlog, End-System Buffer Size, Compression Algorithm

I. Introduction

The Original Equipment Manufacturers (OEM) develop even more complex and efficient embedded systems dedicated to modern aircraft as air traffic is growing on a global scale. Aeronautical equipment enforce a set of sensors, data processing units and actuators, and are strategically distributed on-board to achieve various avionic tasks. Equipment need to exchange a rising amount of data as the complexity and the number of interconnected hardware and software systems grow up. Avionic Full-DupleX switched Ethernet (AFDX) networks address the bandwidth and reliability needs of modern equipment with a high speed and deterministic transmissions. Thus, the network infrastructure and the network source and reception terminals called *End Systems* require a careful analysis to ensure an optimal compliance with aeronautical constraints.

A reception End System (ES) processes the received frames from the network to deliver data to software partitions. However, the overall network configuration implies that a reception ES is supposed to receive frames from several source ESs via Virtual Links (VLs). As a result, the workload of the physical link between the last switch of the network and a reception ES can sporadically be high due to the frame traffic on the VLs. In addition, processing delays occur in the reception ES mostly when the frame passes through the IP and UDP layers. As AFDX networks fall under the scope of the ARINC 664 p.7 standard [1], both partial and total frame loss is prohibited which imposes a queuing (FIFO) reception buffer to store the pending frames. The reception buffer dimensioning is then a critical issue.

The dimensioning of the reception buffer of the ES has received little attention in the literature as the widely-used solution in industry is to oversize the allocated memory to the frames reception. It results in a significant waste of memory resources if we consider the hundreds of reception ESs of the network. Baga *et al.* [2], [3] have proposed a worst-case method based on the characterization of the input frames flow to upper bound the size of the reception buffer while ensuring the lossless storage of all received frames. They have pointed out that the longest sequence of back-to-back frames leads to the Worst Frames Backlog (WFB) in the reception buffer, and the buffer dimensioning relies on the measure of the WFB.

In this paper, we propose to go further in the reduction of the ES reception buffer using a compression method based on an AFDX-adapted Lempel-Ziv-Welch (LZW) algorithm [4], [5] within the ES architecture. The LZW encoder based on a static directory compresses frames to free more memory space of the reception buffer. To the best of our knowledge, it is the first time that a lossless compression algorithm is implemented in the AFDX network context.

In brief, we propose the following contributions to reduce the ES reception buffer size:

- Frames encoding based on the low-cost and AFDX compliant LZW algorithm.
- An hardware implementation and a test platform including a frames generator to assess the memory gain of our method.

The rest of the paper is organized as follow. Section II

978-1-5386-0365-9/17/$31.00 ©2017 IEEE

presents main features of AFDX network and the hardware architecture of the reception ES. Different lossless methods of data compression are criticized and we introduce the LZW algorithm in Section III. Section IV precises the method to generate realistic frames and details the hardware platform developed to measure the memory gain. The simulation results are shown in Section V, we evaluate the memory gain from different frames set, the hardware required resources and the decompression time. Finally, Section VI concludes this paper.

II. AFDX Network : Main Features

The general conception of an AFDX network is presented through a minimal example. Then, we introduce the causes of the Sequences of Back-to-back Frames (SBF) occurrences, and we focus on the ES architecture and the reception buffer.

A. Overview of the AFDX Network & Virtual Links

AFDX networks result in the implementation of two standards: the ARINC 664 [1] which imposes the determinism, the redundancy and the full duplex feature, and the IEEE 802.3 which details the Ethernet protocol and software layers. Determinism is ensured both by the spatial segregation of communication channels with VLs, and by the temporal regulation of source flows with the notion of Bandwidth Allocation Gap (BAG).

The BAG parameter characterizes each VL as the minimal duration between two consecutive frames, but it does not constraint the maximal duration in such a way that frames transmission on a VL is not perforce periodical. To maximize the network workload and to show up the WFB, we will assume that all source ESs send frames onto their VL after each BAG period.

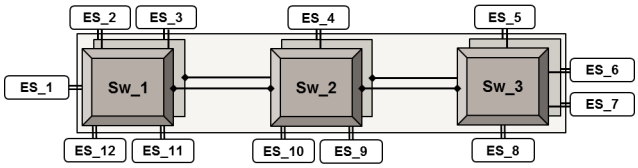


Figure 1: Example of AFDX Network.

The AFDX infrastructure consists in an AFDX interconnect and several source and reception ESs which provide a reliable service of data transmission between avionic systems. The AFDX interconnect is composed of 2 redundant sub-networks including switches and full duplex physical wires. Fig.1 shows an homogeneous AFDX network with 3 redundant switches (Sw_1, Sw_2 & Sw_3) and 12 ESs. Switches routes frames through the network with a *store and forward* policy and according to a static routing table. ESs are physically connected to only one switch with a wire. ESs use static VLs to send frames through unidirectional communication paths which ensure a temporal isolation and a deterministic data transmission. Therefore, the source ESs can instantiate independent unicast or multicast communications to respectively one or several reception ESs.

B. Frame Transmission Delays

A source ES transmits frames to the network respecting with the BAG parameter of its VLs without consideration for the transmission time of other source ESs constituting the network. In other words, the source ESs are asynchronous and transmit frames independently of each other. Regarding the frame transmission delay, an unfavorable scenario is when all source ESs send frames at the same time [6]. It results in a sporadic rise of the network workload [2]. If we assume that a part of these frames are inteded to the same reception ES, a congestion could occur in one of the switch outputs. This leads to the occurrence of a SBF between the final switch and the reception ES.

In addition, the variability of the frame flow is an other source of SBFs. This variability is mainly due to transmission jitters and switch delays [1]. As a source ES generally manages several VLs, a transmission jitter occurs for a frame transmission when a VL requests the access to the physical link while a concurrent VL is transmitting on the physical link. To avoid conflicts, a scheduler included in the source ES multiplexes accesses and it delays concurrent frames to send them one after the other.

Switches induce latencies on the frames re-transmission because of the filtering, policing, switching and monitoring functions that the switch has to lead on each received frame. Moreover, concurrent accesses to switch output ports are the cause of local frame congestions, and so sporadic latencies. These two sources of latencies lead to end-to-end delay variations which are upper bounded and strongly correlated to the network workload. Depending on the way the frames are delayed, the reception ES receives a SBF and it has to process them without loss of data according to ARINC 664 requirements.

C. Architecture of the Reception End System

An End System is an interface to allow communications between avionic equipment via the AFDX network. It enforces the AFDX Ethernet-based protocol and is shared in two isolated parts: one is dedicated to the frames transmission under BAG constraints, and the other is dedicated to received frames to extract data intended to software partitions. In this paper, we focus on the reception part of the ES and we do not consider the transmission. Fig.2 represents the model of a reception ES and the different operations from a frame received from the network to software applications.

A frame transit via a physical wire as a series of bits to at speed of 100 Mbps, and they reach the ES through ARINC 600 connectors. Next, the frame has to pass through the three computing layers of the reception ES: MAC + PHY, AFDX special features and the UDP + IP layer. First, the incoming bits flow has to be de-serialized by the PHY components, and the MAC layer mainly check the address validity regarding to the Configuration Reception Table of the Reception End System (CTRES), the Cyclic Redundancy Code (CRC) validity, and the frame length. If the frame passes these tests, the Integrity Checking (IC) block checks the IC

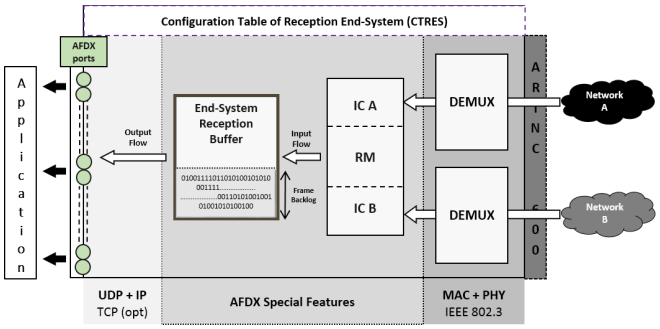


Figure 2: Model of a Reception and Representation of the Backlog in the Reception Buffer.

number considering the VL to which the frame belongs, and the Redundancy Manager (RM) deletes the frame redundancy in the ADFX special features layer. The principle is that the first valid frame reaching the RM is accepted and stored in the reception buffer, while the other frame is discarded. The checks led by the UDP + IP layer are standardized in Ethernet protocol [7], whereas, the software mechanisms will not be specified in this study due to industrial confidential issues.

SBFs are received at wire-speed till the reception buffer. However, the complexity of the UDP + IP checks implies a traffic slowdown and so the need to store the incoming frames to guarantee the non-loss of frame data. In this perspective, the reception buffer is strategically placed at the interface of the low-level hardware chain and the high-level software layer, which is a bottleneck of the ES due to the time consuming operations of the UDP + IP layer. This induces a frame backlog in the reception buffer and it rises the issue of the buffer dimensioning. Thus, we propose to implement a compression algorithm to reduce the required memory space with hardware components.

D. Characterization of AFDX Frames

Before choosing a convenient compression method for AFDX frames, the first step is to represent the information forming the frame. The ARINC 664 p.7 standard indicates that the basic format of a frame is the bit, and a frame is composed of several hundreds of bits, as the maximal frame length is 12,144 bits (1518×8). To ease the representation of frames, we propose the use of hexadecimal numbers (base 16) for a 4-bits packing of the frames. In the compression field, hexadecimal numbers are commonly used to illustrate compression principles [8] because it is more human-friendly than the binary representation. One hexadecimal number will be called a source symbol in the following.

Next, we analyze the constitution of AFDX frames and we assess the potential similarities between frames. As we do not dispose of statistics about the distribution of source symbols inside the AFDX frames based on real industrial frames, we make some assumptions based on the ARINC 664 and the frame fields.

1) Uniform Distribution: A uniform distribution of symbols constituting a frame is the method with the highest source entropy. In other words, a frame is built from a series of uniform draws of source symbols among the hexadecimal numbers, and the probability to draw one given symbol is of $1/16$. Thus, a large number of draws leads to source symbol occurrences quite close, with no a priori on the order of sources symbols in the frame. Uniform distribution is the most general way to create AFDX frames, when statistics about source symbols repartition are not accessible. The disadvantage of the uniform distribution is that the entropy is maximal, and so the compression gain is generally small or even negative, whatever the compression method adopted.

2) Non-Uniform Distribution: A non-uniform distribution offers a reduced source entropy compared to a uniform distribution, and so the potential compression gain increases when the redundancy of source symbols is growing. From this statement, we have to establish if a non-uniform distribution is applicable in the context of AFDX frames. To do that, we have to highlight redundancy both in terms of symbols and sequences of symbols between frames received at a given reception ES.

The analysis of the frames structure underscores some similarities between frames especially on the frames headers due to identical protocol parameters (Fig. 3). As AFDX frames have generally a size less than 300 bytes and the total size of headers is of 47 bytes [1], the headers constitutes a substantial part of the frame. Moreover, a reception ES processes frames from the same source ESs because the network configuration is statically defined, which exacerbates the redundancy in particular for the frames belonging to the same VL. In the same idea, the configuration in reception can be completely opposite in terms of number of VLs and their associated BAG and maximal frame length from one ES to another. Thus, a small number of VLs implies more redundancy of source symbols than a high number of VLs. At last, the payload of AFDX frames includes also a certain degree of redundancy, and it seems reasonable to assume that the nature of data exchanged on a VL remains the same during network operations.

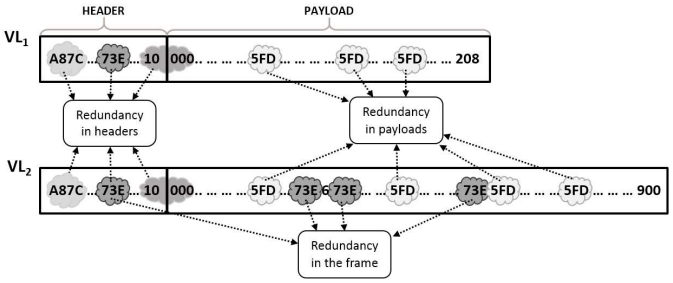


Figure 3: Redundancy of Sequences of Source Symbols in a Frame, and between 2 Frames in the Headers and in the Payloads.

Thereby, redundancy of source symbols can be identified

both on the frame headers and on the frame payloads, and the previous points tend to justify the use of a non-uniform distribution of source symbols. However, the gap between an uniform distribution and a non-uniform distribution is difficult to assess with statistical criteria, but we assume that the distribution that we use in this study is *relatively* close to an uniform distribution. We generate frames from uniform draws in a finite number of lists which contain random sequences of source symbols with a length between 1 and 4 symbols, and we dispose of weighting coefficients to adjust the probabilities to draw a sequence in the lists. In this way, the higher the weighting coefficients are for the list of long sequences, the higher the redundancy is.

III. Reduction of the Reception Buffer Size

In this section, we present some lossless algorithms among the most widely used and we study their applicability in the AFDX context. Next, we highlight the LZW algorithm and the way that it is adapted to the ES to compress incoming frames.

A. Lossless Compression Algorithms

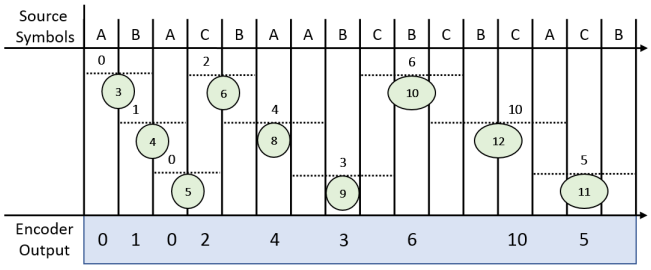
The general principle of compression is to reduce the number of necessary bits to store a piece of information. A prime feature classifies compression algorithms in 2 families: those who lose data during compression called lossy algorithms, and those who exhaustively keep data during compression called lossless algorithms. Lossy compression is acceptable when a part of the source information can be lost with a valuable impact on the returned quality like in image [9] or sound processing. However, the critical context of AFDX networks forbids the loss of frame bit data because a tiny modification in frame bits could have unexpected and potentially catastrophic effects on the passengers safety. For this reason, we only consider the lossless compression algorithms to respect the frames integrity. The lossless compression algorithms are widely used in data communication systems to reduce the amount of bits transferred.

The entropy coding family is a part of lossless compression codes and is based on the principle that a change in the source data representation allows data compression. To do that, entropy coding relies on statistics of the source symbols occurrences to encode new words, whose length depends on the number of these occurrences: the more frequently a source symbol (or a sequence of source symbols) appears, the shorter its encoded word is. Therefore, the resulting codes are often variable-length like Golomb-Rice [10], [11] coding or Huffman coding [12].

Golomb-Rice coding consists in dividing positive integer N (the source symbol we want to encode) by a parameter m such as: $N = q \times m + r$, where q is the quotient, and r the binary rest. Efficiently of Golomb-Rice coding strongly depends on the value of m which the optimal value itself depends on the distribution of source symbols. Huffman coding is based on the construction of a source symbols occurrences probability tree where each node is associated to the occurrence of a symbol. Huffman coding is optimal when the source is static and the distribution of source symbols is known before compression. However, the variability of AFDX flows does not permit the use of such methods for the frames reception because Golomb-Rice and Huffman encodings are only effective if the distribution is known in advance. A wrong setting of these algorithms parameters could lead to an increase of the number of bits to encode the source symbols, and so a rise of the required memory space.

B. LZW Algorithms

Proposed in 1984, LZW algorithm result of many improvements based on the works of Lempel, Ziv [4] and Welch [5]. LZW algorithms are relatively light, easy to implement both in low-level systems [13] and in high-level systems [14] and they do not require an analysis of source symbols before compression. LZW algorithms and their derivatives (LZ77, LZ4, LZMW, LZAP, LZJB, etc...) are widely used especially in file formats like PDF, GIF or TIFF.



	Dictionary Addresses	Source Symbols			
0	0000	A			
1	0001	B			
2	0010	C			
3	0011	A	B		
4	0100	B	A		
5	0101	A	C		
6	0110	C	B		
7	0111	...	...		
8	1000	B	A	A	
9	1001	A	B	C	
10	1010	C	B	C	
11	1011	A	C	B	
12	1100	C	B	C	A
13	1101	...	...	...	...
14	1110	...	...	...	...
15	1111	...	...	...	...

Dictionary 1 (0–2) · Dictionary 2 (3–7) · Dictionary 3 (8–11) · Dictionary 4 (12–15)

Figure 4: Example of the LZW Algorithm with 3 Sources Symbols and 4 Dictionaries.

The principle of LZW algorithms is to complete an encoding table called dictionary which encodes sequences of source symbols in short fix-length words corresponding to the respective dictionary addresses where the sequences are stored. Fig. 4 illustrates the compression mechanisms through a simplified example of the LZW algorithm used in this paper. The three source symbols (**A**, **B** and **C**) are coded on 4 bits like the hexadecimal base that we use to represent frames content. The received sequence is **ABACBAABCBCBCACB**. The dictionary extends on a 4-bit address range or 16 possible addresses and contains 4 smaller dictionaries: the dictionary 1 (addresses 0 to 2) includes the sequences of 1 symbols (so the initial alphabet), the dictionary 2 (addresses 3 to 7) includes the sequences of 2 symbols and so on until the dictionary 4 (addresses 12 to 15) [15].

At the first step, the encoder receives the symbol **A**. The encoder memorizes this symbol as it already exists in the dictionary 1, and the word **0000** is sent to the encoder output. The next symbol is **B**. The sequence **AB** has not yet been encountered, so the sequence **AB** is stored in the dictionary 2 associated to the word **0011** and the word **0001** is sent to the encoder output. These steps are repeated until all the available dictionaries addresses are associated to a sequence of source symbols. Thus, the next time when a memorized sequence is encountered, the corresponding word is directly placed in output.

The memory gain comes from the difference between the number of bits used to encode the source symbols and the output words. In this example, a source symbol is encoded on 4 bits, so 2 source symbols represent 8 bits. Whereas, a word is encoded on 4 bits. Therefore, if a sequence of 2 symbols is compressed, the gain is of $(8-4)/4 \times 100 = 50\%$. The higher the number of symbols in a sequence is, the more important the memory gain is.

C. Hardware Implementation of LZW Algorithm

We now propose an adaptation of the previous LZW algorithm to compress the frames stored in the ES reception buffer. The LZW algorithm is constituted of one encoder and one decoder respectively placed before and after the reception buffer. To ensure fast operations, we implement the two blocks in hardware into a standard FPGA.

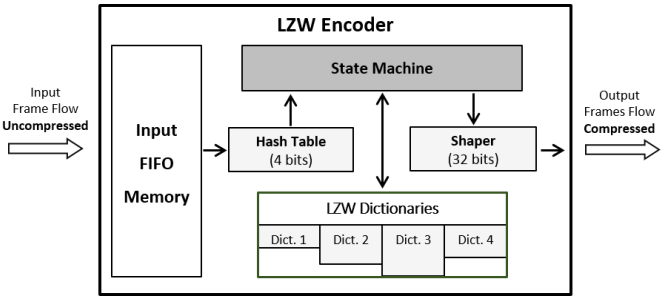


Figure 5: Block Representation of the LZW Encoder Implementation.

Fig. 5 represents the hardware architecture of the LZW encoder inspired by the works of Ce *et al.* [16]. The input frames flow reaches the input of the LZW encoder by a 32-bit width communication bus, and is stored in a FIFO memory. Then, the hash table extracts source symbols of 4 bits. The LZW dictionary stores the sequences of source symbols and generates the n-bits output words. Finally, the output shaper concatenates the n-bit output words to send a 32-bit word to the reception buffer via a communication bus.

The encoded words are n-bit width which relies on the number of sequences that the 4 dictionaries can store. Generally speaking, larger directories allow a better frames compression, but it implies a higher resource memory need and a higher decompression time cost. The total dictionary size requires a tradeoff between the compression gain and the memory resources usage to be effective. Moreover, the issue of the memory repartition between the 4 dictionaries have to be analyzed because the number of sequences stored in each dictionary has an important impact on the final compression gain.

IV. Simulations & Results

The compression gain assessment of the AFDX frames with our hardware LZW compressor requires a complete hardware platform that we present in this part. Finally, we measure compression gains for a set of frames and we discuss about the optimal way to share the memory space among the 4 dictionaries.

A. A Custom made Hardware Testing Platform

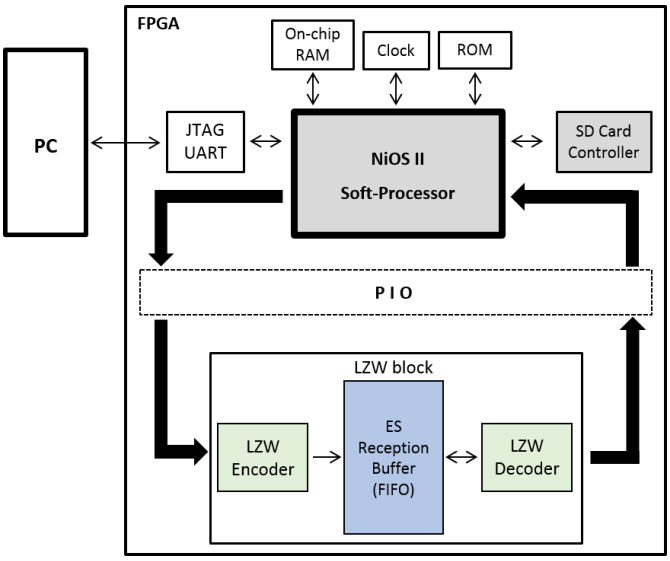


Figure 6: Hardware Test Platform to Evaluate the Compression Gain

Fig. 6 represent our hardware testing platform implemented in an FPGA. To realize a compression gain measure, we need a set of frames composed of several millions of source symbols. These sets are created outline with a configurable

frames generator coded in C. Thus, we choose the degree of redundancy that we want to include in the source symbols. A reference set of frames is drawn and a series of secondary sets are then created by modifying the order of frames. The order that the frames are received changes the sequences stored into the dictionaries, and so the potential memory gain. The reference set and the secondary sets are recorded in separate files in a SD card placed onto the FPGA board.

The hardware testing platform is built around a NiOS II soft-processor which sends frames to the LZW block via a Parallel Input Output (PIO) from the SD card. The compressed frames are recorded in the reception buffer. The NiOS II can either read the compressed frames directly to assess the compression gain by comparing with the size of the initial set of frames, or read the compressed frames via the LZW decoder to control the duration of the LZW decompression and the frames validity. The results are assembled in a report sent to the PC via the JTAG UART.

B. Compression Gain Measures

1) Gain for a standard set of frames: Firstly, we create a standard set of frames and we analyze the compression gain for different dictionaries configurations. Fig. 7 represents the minimum, average and maximum gain obtained when the order of frames is changed, as a function of the size of the dictionary 4. The graphs **a**, **b**, **c** and **d** respectively corresponds to 9, 10, 11 and 12 bits to encode the encoder output words. Under 9 bits, the compression gain is negative. From 9 bits and more, it appears that the dictionary 2 has to be systematically set at the maximal size or 256 possible sequences. Therefore, the remaining address range is split between the dictionaries 3 and 4 following this equation : $S_3 + S_4 = 2^n - 256 - 16$, where S_3 and S_4 are respectively the size of dictionaries 3 and 4.

We obtain similar concave shapes for the compression gain, and it appears that when the dictionary 4 is too small the gain decreases. Then, the gain reaches a maximum and decreases again when the size becomes too large until being negative for all number of encoding bits. The best gain is obtained for a 10-bit encoding configuration (Fig. 7b) with 22.35% but the size of dictionary 4 has to be precisely set. However, for 11 and 12 bits (Fig. 7c,d), the gain is lower but the range of values where the gain is almost constant is larger than for 10 bits. For 9 bits (Fig. 7a), the maximal gain is lower than for 10 bits and it follows a bell shape with a maximum of 8.59%.

To sum up, when the size of dictionary 4 is small, the gain is small too. Then, a range of values where the gain is constant appear, and the higher the number of encoding bits is, the more larger this range of values is. Lastly, the gain decreases from a certain size of the dictionary 4 until being negative. The rise of the gain corresponds to the dictionary 4 growth which allows the highest compression on a given sequence of source symbols. Conversely, when the dictionary 4 becomes too large, the dictionary 3 is no longer of sufficient size to allow frequent recording of sequences of 4 source symbols. The constant part of the gain curve results to an area

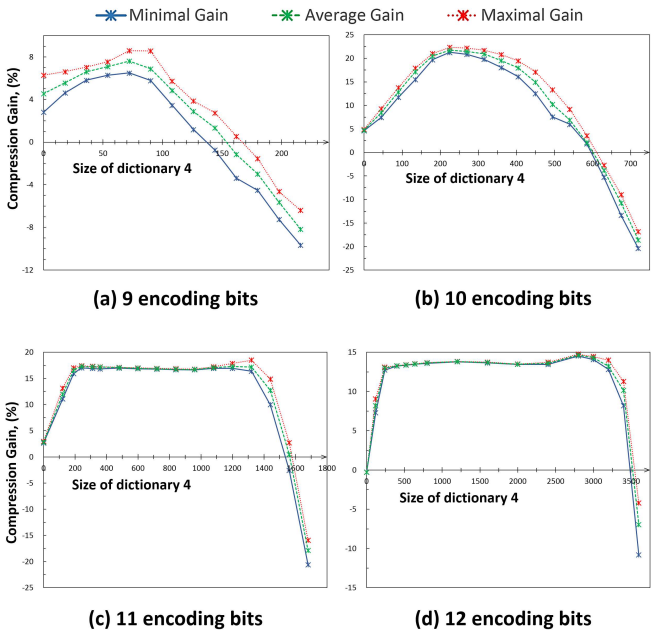


Figure 7: Compression Gain for 1 Reference Set and 20 secondary Sets of Frames for Different Number of Encoding Bits as a function of the size of the dictionary 4.

where the size of dictionaries 3 and 4 is adequate. From these observations, we make the following hypothesis: the optimal dictionary configuration is obtained when both dictionary 3 and 4 are enough big and with the smallest encoding size allowing this configuration.

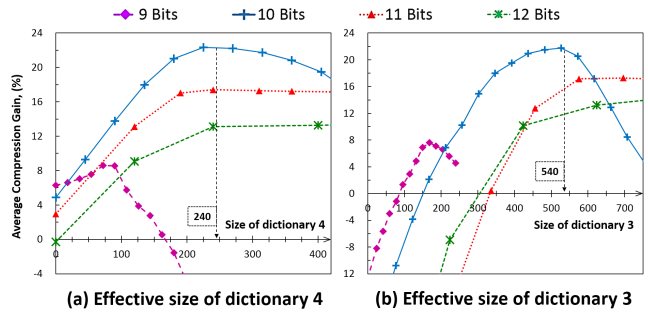


Figure 8: Compression Gain for 1 Reference Set and 20 secondary Sets of Frames for Different Number of Encoding Bits as a function of the size of the dictionary 4 (a) and the size of the dictionary 3 (b).

To verify this hypothesis, we plot the average compression gain as a function of the dictionary 4 size (8a) and the dictionary 3 (8b) for the four encoding sizes previously highlighted. We focus on the critical areas of the Fig. 7 which corresponds to the increase part and the decrease part.

Fig. 8 provides expected values compared to Fig. 7. We observe the same behavior: the gain starts negative and increases proportionally with the size of dictionaries 3 and 4,

then it quickly becomes constant for 11 and 12 encoding bits. The large constant area for 11 and 12 encoding bits corresponds to the space where the dictionaries 3 and 4 are wide enough but not leading to the optimal gain because of the number of required encoding bits is too high. However, for 10 encoding bits, the constant area is narrow and corresponds to the maximum gain for the set of frames. For 9 encoding bits, the gain has not a constant area. The gain behavior for 9 and 10 encoding bits is caused by the limited size of dictionaries 3 and 4. These results show that the compression requires a minimum size for the dictionary 3 before considering the memory space of the dictionary 4. The sequences of source symbols stored in the dictionary 4 have to derive from a parent sequence of the dictionary 3. In the same way, the dictionary 3 has parents on the dictionary 2. The heredity between the dictionaries imposes to the dictionary 2 the maximum size of 256 sequences of 2 symbols which cover all possible combinations.

Table I: Hardware Resources for 10 Encoding Bits (Best Gain Measured: 22.35%).

Logic Elements	19,625	236,364
Registers	4,742	12,266

From the Fig. 8, we note that the compression gain for 10, 11 and 12 encoding bits starts to be constant for a size of 240 for the dictionary 4 and 540 for the dictionary 3. For 10 encoding bits, the gain offers a short constant area because the minimum size required both for the dictionaries 3 and 4 is almost unique, and the gain reaches its maximal value of 22.35% for the resources cost precised in Tab. I. For 9 encoding bits, the configuration is not relevant because the sizes of the dictionaries 3 and 4 are not wide enough, and the resulting gains are small.

2) Gain as a function of the memory space occupied by the Dictionary 4: On the Fig. 9, 10 and 11, we represent the compression gain as a function of the part of memory space occupied by the dictionary 4 on the remaining space allocated for the dictionaries 3 and 4 for three sets of frames coming from different generation parameters. A relative part (in %) allows to represent several encoding sizes in the same graph. The advantage of the graphs is that they are more summary as the gain is measured for all the values of dictionaries.

Fig. 9 is obtained from the first standard set of frames. Fig. 10 is built from a set of frames less dictionary 4 oriented, and Fig. 11 is built from wider generation lists and dictionary 4 oriented.

On the Fig. 10, the maximum gain of 12.0% is for 11 encoding bits and the average gain is better than the gain obtained for 10 encoding bits. These results are due to the fact that the dictionary 4 has to have the same size than for the standard set of frames, but at the same time, the dictionary 3 has to be wider because the frames generation is more short sequences-oriented. Therefore, the result of the addition of the

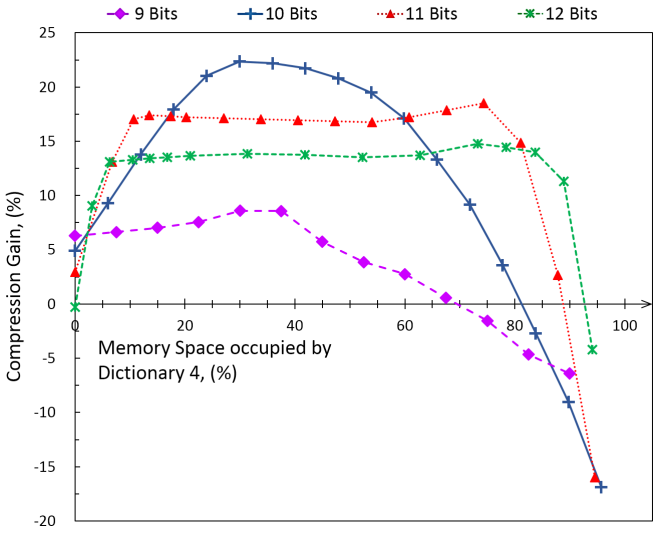


Figure 9: Compression gain for a standard set of frames (12 millions of source symbols).

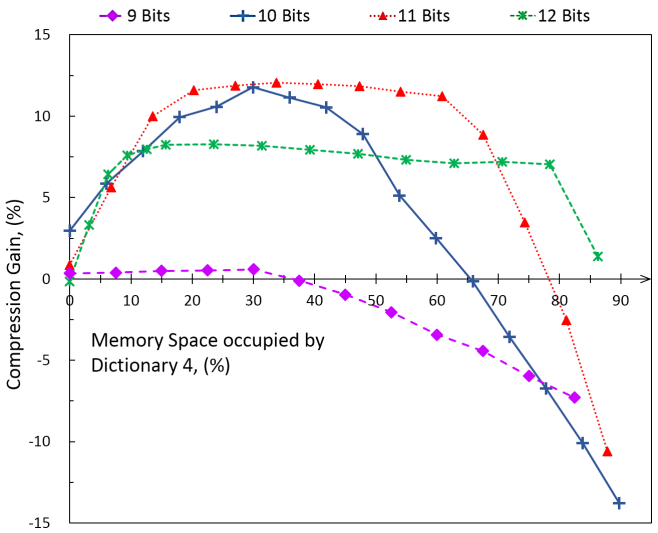


Figure 10: Compression gain oriented to short sequences.

dictionaries 3 and 4 has to be bigger than the previous setting, and so 10 encoding bits is not the best solution.

On the Fig. 11, the curve corresponding to 9 encoding bits is not plotted because the gain remains negative in this configuration. The compression gain for 10 encoding bits is low, and 12 encoding bits offer a better compression gain than in other examples. The best encoding configuration is again for 11 bits (13.5%) although the constant area is smaller. Since the generation lists are wider to ensure more random draws, the dictionaries have to be wider to guarantee the most part of the sequences ends up inside the dictionaries 3 and 4, and so the required size of dictionaries favors high values for encoding bits. Hence, we can establish a link between redundancy of AFDX frames and the optimal sizes of the dictionaries.

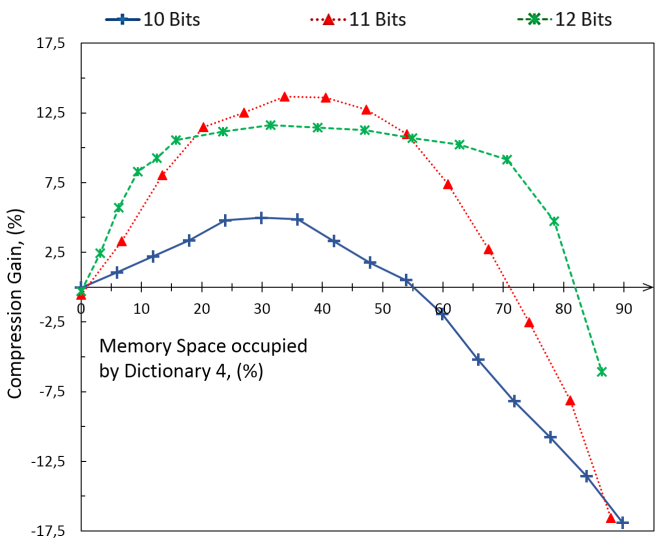


Figure 11: Compression gain for a dictionary 4 oriented set of frames with more randomness draws than the standard set of frames.

V. Conclusion and Perspectives

An LZW compression method is applied to an ES reception buffer, and the compression gain is assessed for variable size of dictionaries and three set of frames, each composed of millions of source symbols. We establish a link between the degree of redundancy in the frame source symbols and the reachable maximal gain. An optimal gain can be obtained if the input frame flow is analyzed off-line, as we show that general settings on the dictionary sizes can not provide high gains.

In future works, an adaptive LZW algorithm could be proposed with variable dictionary sizes according to statistics computed on the input frame flow. After a certain number of iterations, the potential obtained gain could be higher than with static dictionaries.

References

[1] A. E. E. Committee, *ARINC specification 664P7: Aircraft Data Network, part 7: Avionics Full-Duplex Switched Ethernet (AFDX) network*. Aeronautical Radio Inc., June 2005.

[2] Y. Baga, F. Ghaffari, E. Zante, M. Nahmiyace, and D. Declercq, "Worst frame backlog estimation in an avionics full-duplex switched ethernet end-system," in *2016 IEEE/AIAA 35th Digital Avionics Systems Conference (DASC)*, pp. 1–10, Sept 2016.

[3] Y. Baga, F. Ghaffari, D. Declercq, E. Zante, and M. Nahmiyace, "Probabilistic model of afdx frames reception for end system backlog assessment," in *2017 IEEE 12th International Symposium on Industrial Embedded Systems (SIES)*, June 2017.

[4] T. A. Welch, "A technique for high-performance data compression," *Computer*, pp. 8–19, 1984.

[5] J. Ziv and A. Lempel, "A universal algorithm for sequential data compression," *IEEE Transactions on Information Theory*, pp. 337–343, 1977.

[6] G. F. R. Coelho and J. L. Scharbarg, "Dimensioning buffers for afdx networks with multiple priorities virtual links," *Digital Avionics Systems Conference (DASC), 2015 IEEE/AIAA 34th*, 2015.

[7] "Internet protocol darpa internet program protocol specification," *RFC 791*, 1981.

[8] S.-J. Kwon, S.-H. Kim, H.-J. Kim, and J.-S. Kim, "Lz4m: A fast compression algorithm for in-memory data," in *2017 IEEE International Conference on Consumer Electronics (ICCE)*, pp. 420–423, Jan 2017.

[9] O. K. Al-Shaykh, I. Moccagatta, and H. Chen, "Jpeg-2000: a new still image compression standard," in *Conference Record of Thirty-Second Asilomar Conference on Signals, Systems and Computers (Cat. No.98CH36284)*, vol. 1, pp. 99–103 vol.1, Nov 1998.

[10] S. Golomb, "Run-length encodings (corresp.)," *IEEE Transactions on Information Theory*, vol. 12, pp. 399–401, Jul 1966.

[11] R. Rice and J. Plaunt, "Adaptive variable-length coding for efficient compression of spacecraft television data," *IEEE Transactions on Communication Technology*, vol. 19, pp. 889–897, December 1971.

[12] D. A. Huffman, "A method for the construction of minimum-redundancy codes," *Proceedings of the IRE*, vol. 40, pp. 1098–1101, Sept 1952.

[13] M. b. Lin, J. f. Lee, and G. E. Jan, "A lossless data compression and decompression algorithm and its hardware architecture," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 14, pp. 925–936, Sept 2006.

[14] J. Zhan, Q. Zhou, S. Bai, L. Cuihua, B. Hu, and L. Li, "Bd-lzw picture compression algorithm for wsn system," in *2008 Third International Conference on Pervasive Computing and Applications*, vol. 1, pp. 146–150, Oct 2008.

[15] M.-B. Lin, "A parallel vlsi architecture for the lzw data compression algorithm," in *Proceedings of Technical Papers. International Symposium on VLSI Technology, Systems, and Applications*, pp. 98–101, June 1997.

[16] Z. Ce and X. Hui, "Design and implementation of lossless compression encoding for high-speed data acquisition and storage," *2015 12th IEEE International Conference on Electronic Measurement & Instruments (ICEMI)*, pp. 502–506, July 2015.