UHF uplink FEC study

AcubeSAT-COM-BT-025

Mavrikakis Parmenion (parismavris@gmail.com)

July 29, 2019
Version: 0.5

Aristotle University of Thessaloniki

Aristotle Space and Aeronautics Team
CubeSat Project

2019

Contents

Changelog

Date	Version	Document Status	Comments
29/07/2019	`0.5`	INTERNALLY RELEASED	Final Version
25/07/2019	`0.4`	DRAFT	Adding convolutional codes
16/07/2019	`0.3`	INTERNALLY RELEASED	
16/07/2019	`0.2`	DRAFT	Adding content
15/07/2019	`0.1`	DRAFT	Initial revision

This is the latest version of this document (`0.5`) as of July 29, 2019. Newer versions might be available. Check `https://helit.org/mm/docList.php`

1 Introduction

Satellite data communication requires the establishment of a reliable, error-controlled channel for enhanced performance. Fortunately, we have the ability to encode our data in order to achieve higher data transmission quality and less bit errors in our messages by using forward error correction coding (FEC). If you want an analytical presentation of encoding and decoding for Golay and Reed-Solomon codes or info about their data rate and complexity you can find it here!

2 Golay error detection and correction code

This code has been named after its creator, Marcel J. E. Golay. Binary Golay code is a linear error-correcting code used in digital communications. There are two types of binary Golay codes:

- Golay (23,12,7),
- Golay (24,12,8)

2.1 Golay (23,12,7)

The numbers (23,12,7) indicate the total bits of every codeword (n=23), its info bits (k=12) and the minimum Hamming distance[1] between two different codewords (d=7). In this case, a Golay codeword has twelve bits of information to which are appended 11 check bits, which are derived from a modulo-2 division. Also, we know that this code can detect and correct **3 bit errors**, in any pattern.

In [1] there is a link level simulation with which we can have an approximation of the function of channel coding.

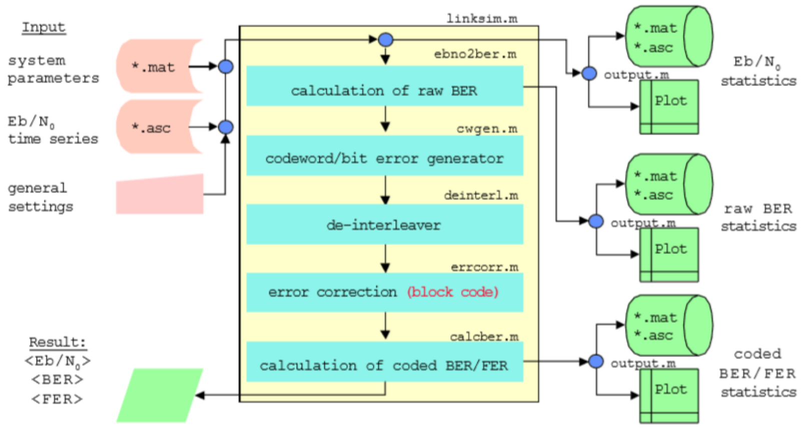


***Figure 1**: Overview of the link level simulation*

[1]The number of bits which differ between two binary numbers

The mean BER can be approximated by assuming for each frame error half of the codeword bits to be in error. Thus $\mathrm{BER}_{\mathrm{b,i}} \approx 0.5 \cdot n \cdot FER_{\mathrm{b,i}}$.

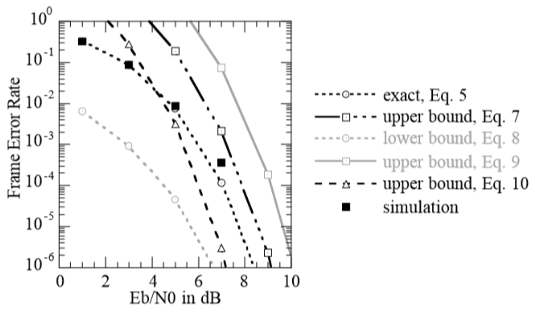


***Figure 2**: FER simulation and analytical bounds for the Golay (23,12) code*

So we have an approximate estimation about the BER/SNR relation for the Golay (23,12) code.

2.2 Golay (24,12,8)

In this case an extra parity bit is added and as a result we have a 3-byte codeword with 12 info bits and minimum Hamming distance d=8.

For error correction only:

- 100% of one to six bit errors detected, any pattern
- 100% of odd bit errors detected, any pattern
- 99.988% of other errors detected

Using error correction:

- 100% of one to three bit errors corrected, any pattern
- 100% of four bit errors detected, any pattern
- 100% of odd numbers of bit errors detected, any pattern
- 0.24% of other errors corrected

This code is a reliable and easy to implement solution (the Golay code could be used in even the smallest micro-controllers, such as the PIC series and 68HC05) and it was used before in missions, for instance UKube-1. Last but not least, Golay code can't correct burst errors (in our case more than 3-bit errors) but we can solve that problem by using interleaving.

In [2] you can find a review on research work presented in the field of FPGA, which aims to present systems which decrease the complexity of systems to accomplish the requirements of low latency data and high speed application. Some of them are Golay based and can be used for implementation or as a start for its design.

3 Reed-Solomon

3.1 General

Reed–Solomon codes are a group of error-correcting codes that were introduced by Irving S. Reed and Gustave Solomon in 1960[3]. They have many applications, the most prominent of which include consumer technologies such as CDs, DVDs, Blu-ray discs, QR codes, data transmission technologies such as DSL and WiMAX, broadcast systems such as satellite communications, DVB and ATSC, and storage systems such as RAID 6. There are two basic types of Reed–Solomon codes, original view and BCH view, with BCH view being the most common as BCH view decoders are faster and require less working storage than original view decoders.

The properties of Reed-Solomon codes make them especially suited to the applications where burst error occurs and they are the following:

- The code disregards how many bits in a symbol are incorrect. If multiple bits in a symbol are corrupted, it only counts as a single error. Alternatively, if a data stream is not characterized by error bursts or drop-outs but by random single bit errors, a Reed-Solomon code is usually a poor choice. More effective codes are available for this case.
- Designers are not required to use the natural sizes of Reed-Solomon code blocks. A technique known as "shortening" produces a smaller code of any desired size from a larger code. For example, the widely used (255,251) code can be converted to a (160,128). At the decoder, the same portion of the block is loaded locally with binary zeroes.
- A Reed–Solomon code operating on 8-bits symbols has $n = 2^8 - 1 = 255$ symbols per block because the number of symbol in the encoded block is $n = 2^m - 1$.

Each symbol consists of m bits. The total number of symbols is n, and the number of information symbols is k. So, there are $m \cdot n$ bits of information to be encoded. The remaining symbols (n-k) are used as parity symbols. For a given symbol width, n has a maximum value of $2^m - 1$. The correction capacity is t and is computed by the equation $2t = n - k$, as shown below.

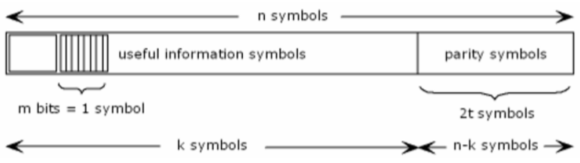


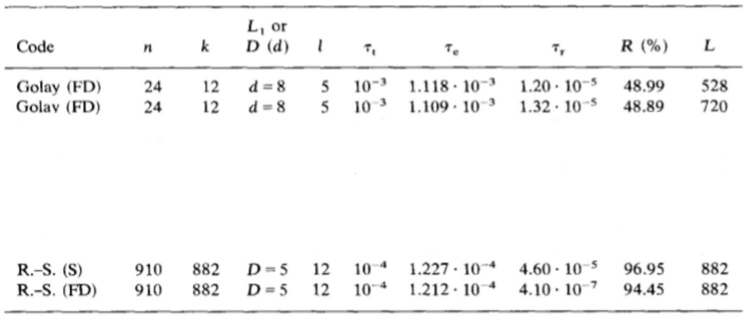
Figure 3: *Reed-Solomon Codeword*

3.2 Introduction to Golay vs. RS

In [4] you can find the presentation of a software program simulating data transmission through a noisy channel. The program can simulate transmissions through a modelization of the error rate and the kinds of errors, from submarine ones (error rate about 10^{-2}) to transmissions via satellites (error rate about 10^{-5} to 10^{-8}). Below there are the results of data transmission and numerical results obtained with Reed-Solomon codes and with the Golay (24,12,8) code.

Notations:

1. τ_e: The experimental binary error rate,
2. τ_t: The theoretical binary error rate,
3. τ_r: The binary residual error rate,
4. R: The transmission efficiency,
5. l: The maximal error length,
6. L: The information frame length,
7. FD: The full duplex,
8. S: The simplex,
9. l_1: The maximum length of correctable bursts,
10. d or D: The minimum distance of the code.



Code	n	k	L_1 or D (d)	l	τ_t	τ_e	τ_r	R (%)	L
Golay (FD)	24	12	$d = 8$	5	10^{-3}	$1.118 \cdot 10^{-3}$	$1.20 \cdot 10^{-5}$	48.99	528
Golav (FD)	24	12	$d = 8$	5	10^{-3}	$1.109 \cdot 10^{-3}$	$1.32 \cdot 10^{-5}$	48.89	720
R.–S. (S)	910	882	$D = 5$	12	10^{-4}	$1.227 \cdot 10^{-4}$	$4.60 \cdot 10^{-5}$	96.95	882
R.–S. (FD)	910	882	$D = 5$	12	10^{-4}	$1.212 \cdot 10^{-4}$	$4.10 \cdot 10^{-7}$	94.45	882

***Figure 4**: Results*

We observe that Golay (24,12,8) has approximately 49% transmission efficiency. In contrast, Reed-Solomon's efficiency is much higher, 94% to 97%, and its binary error rate is lower for the same initial data and channel. Also, both of them can correct enough errors in order to have a really good result, but Golay needs much less length to achieve it. The errors might be too many for satellite communication but this simulation gives us an idea about the difference between the two codes.

3.3 Different RS codes performance

Although in [5] it is considered that the frequency range uplink is 136-146 MHz, we can compare two Reed-Solomon coding schemes. Before that, we should remember that QPSK/OQPSK has been previously selected and that the comparison is between coded and uncoded QPSK / OQPSK in Rician & Log-normal channel. So, in figure 5 we notice that BERs of RS (255,243) and RS (12,9) are approximately identical while $SNR \leq 12dB$. But, when $SNR > 12dB$, then RS (255,243) code has much smaller BER as the SNR increases. On the contrary, in figure 6 the BERs remain almost identical with RS (255,243) being better than RS (12,9) for $SNR < 12dB$ and the other way around for $SNR > 14dB$. Obviously, the combination of RS (255,243) code with OQPSK is preferred for the uplink under these circumstances (light fading channel conditions) because the SNR needed for BER$\approx 10^{-5}$ is much lower (but still too high and even higher for BER$\approx 10^{-6}$).

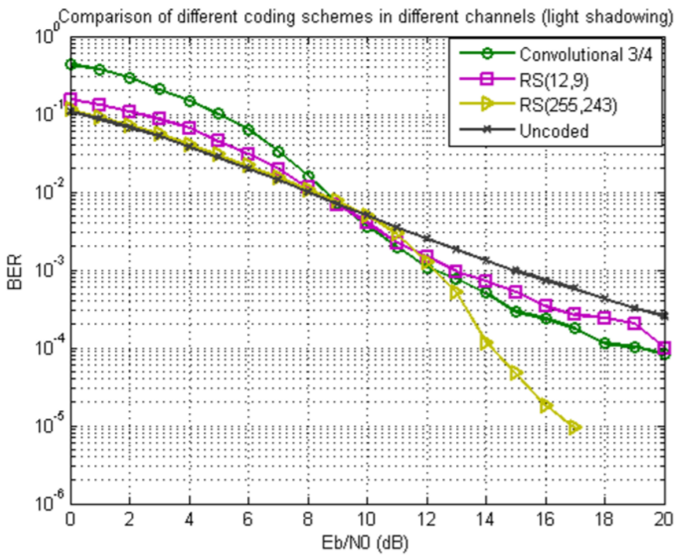


Figure 5: *Comparison of coded and uncoded QPSK / OQPSK for different coding schemes in Rician & Log-normal channel (light shadowing $K = 4$, $\mu = 0.13$ and $\sigma = 1.0$).*

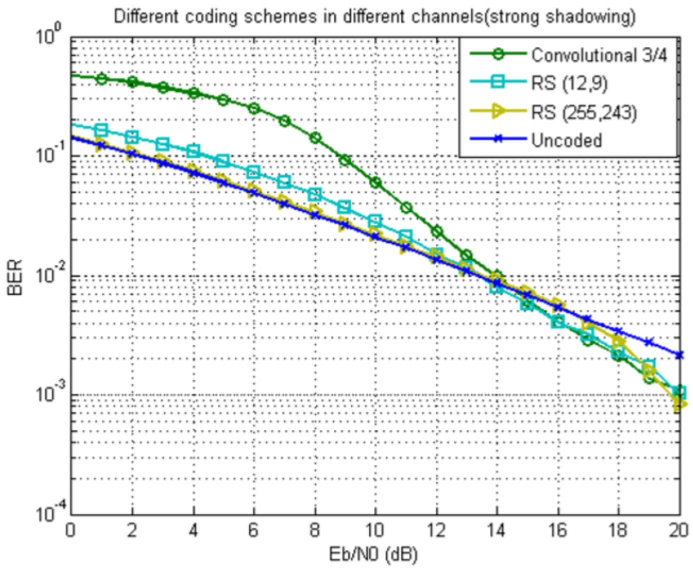


Figure 6: *Comparison of coded and uncoded QPSK / OQPSK for different coding schemes in Rician & Log-normal channel (strong shadowing: $K = 0.6$, $\mu = -1.08$ and $\sigma = 2.5$)*

3.4 MSK RS-Golay comparison

Figure 7 depicts the comparison between uncoded and coded MSK (we are probably going to use it for the uplink) scheme with several coding techniques. We see that Reed-Solomon (511,479,33) code outperforms the Golay and Hamming codes as the SNR increases ($SNR > 6dB$). Moreover, Reed-Solomon code gives the second highest gain[2] and results in a better BER. You can find the whole presentation and analysis in

[2]The difference in required SNR to obtain a certain BER for a specific code compared to uncoded system (to obtain higher gain a long code must be used, this demands more complex decoder and high energy consumption).

[6].

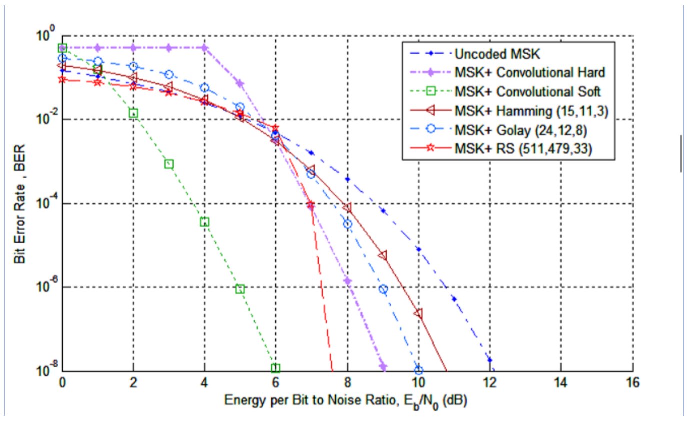


***Figure 7**: Bit error rate analysis of various coding techniques*

Figure 8 presents the characteristics of different Reed-Solomon codes.

Figure 9 presents the performance of Reed-Solomon codes with various rates. The coding gains of different Reed-Solomon codes are not the same for all the BERs.

We observe that as the information length of Reed-Solomon codes increases, the gain decreases.

Parameters of various RS codes				
Codeword length: n	**Information length: k**	**Check bytes: n-k**	**Correcting Capability: (n-k)/2**	**Rate of Code: R=k/n**
512	508	4	2	0.994
512	504	8	4	0.986
512	496	16	8	0.970
512	480	32	16	0.939

***Figure 8**: Characteristics of RS codes presented in Fig. 10*

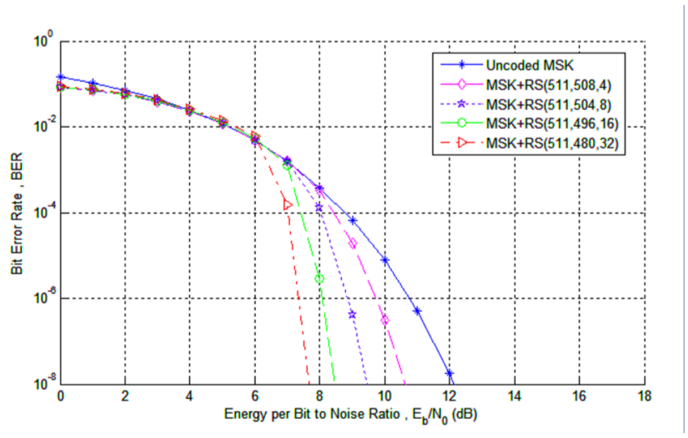


***Figure 9**: Performance Evaluation of RS code, MSK (AWGN)*

By comparing figures 7 and 9 we can observe that Golay (24,12,8) is better that RS (511,508,4) in terms of BER and gain, but it always has 0.5 rate.

Reference [7] shows that by employing the algebraic approach in the design of RS algorithms, the final FPGA implementation takes less area than the traditional implementations. So, this can be a useful tactic for Reed-Solomon implementation.

4 BCH (63,56) for satellite communication

4.1 BCH in general

The name Bose–Chaudhuri–Hocquenghem (BCH) arises from the initials of the inventors' surnames. BCH codes operate over finite fields or Galois fields. BCH codes form a large class of powerful random error-correcting cyclic codes. This class of codes is a remarkable generalization of the Hamming code for multiple-error correction. BCH coding can be explained in the following steps:

For any positive integers $m \geq 3$ and $t \leq 2^{m-1}$, there exists a primitive BCH code with the following parameters:

- Block length: $n = 2^m - 1$
- Number of parity-check digits: $n - k \leq m \cdot t$
- Minimum distance: $d_{min} \geq 2 \cdot t + 1$

We call this code a t-error-correcting BCH (n,k) code. The code operates in binary Galois Field GF (2^m), where m is the number of information bits. Any operation on elements of GF always results in another field element. The generator polynomial is:

$$g(x) = 1 + g_1 \cdot x + g_2 \cdot x^2 + \cdots + g_{n-k-1} \cdot x^{n-k-1} + x^{n-k} \tag{1}$$

With respect to satellite communication BCH codes are well suited thanks to:

- Well understood algebraic structure,
- Good hamming distance for given information and code length,
- Easy implemention in hardware.

4.2 CCSDS BCH (63,56)

As presented in **CCSDS TC SYNCHRONIZATION AND CHANNEL CODING**:

4.2.1 Codeword Format

The BCH codeword format is a fixed-length data entity shown in figure 10. The codeword is formulated using a systematic coding technique which contains 56 information bits in the leading octets, and the error control bits in the last octet. The BCH codeword contains an integer number of octets with an overall length of 8 octets (64 bits).

The complements of the seven parity check bits, P_0 through P_6, are located in the first seven bits of the last octet of the BCH codeword. The complements are used to

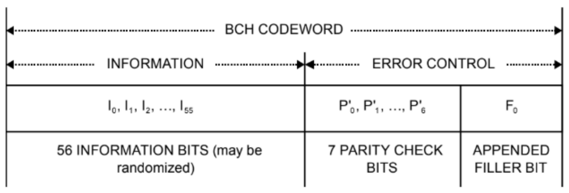


Figure 10: *Codeword Structure*

aid in maintaining bit synchronization and detection of bit slippage. The last bit of the last octet, F_0, is a filler bit appended to provide an overall codeword length which is an integer number of octets. This filler bit shall always be a zero.

4.2.2 Encoding

A systematic block coding procedure shall be used which always generates 7 parity check bits per codeword and which is always computed from 56 information bits. The parity check bits are then complemented and placed into the codeword as shown in figure 11. The code used is a (63,56) modified Bose-Chaudhuri-Hocquenghem (BCH) code which uses the following generator polynomial to produce the seven parity bits:

$$g(x) = x^7 + x^6 + x^2 + 1 \tag{2}$$

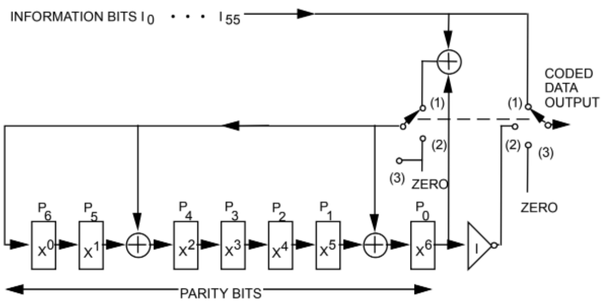


Figure 11: *(63,56) Modified BCH Code Generator*

4.2.3 Fill data

If the Transfer Frame(s) to be transmitted in a CLTU do not fit exactly within an integral number of BCH codewords, then filler bits shall be appended to the last Transfer Frame to be transmitted in the CLTU until an integral number of BCH codewords is completed. The pattern of the fill shall consist of a sequence of alternating ones and zeros, starting with a zero. If randomization is used, the fill data mentioned above shall be added either before or after randomization.

4.2.4 Decoding

Codewords that have been encoded using the modified BCH code described in 4.2.2 may be decoded either in an error-detecting mode (Triple Error Detection, or TED) or in an error-correcting mode (Single Error Correction or SEC), depending on mission requirements. When the error-detecting mode is chosen, one, two or three bits in error will be detected within the codeword (not counting the appended Filler Bit); when the error-correcting mode is chosen, one bit in error will be corrected and two bits in error will be detected.

4.2.5 BCH (63, 56) performance

Currently, at the receiver side, hard decision is taken on the received symbols. The codeword error rate (CER) performances of the hard decision decoded BCH (63, 56) code on the AWGN channel are shown in Figure 12. They are rather poor, due to the very limited error correction capability (SEC mode is examined because this is specified by ECSS).

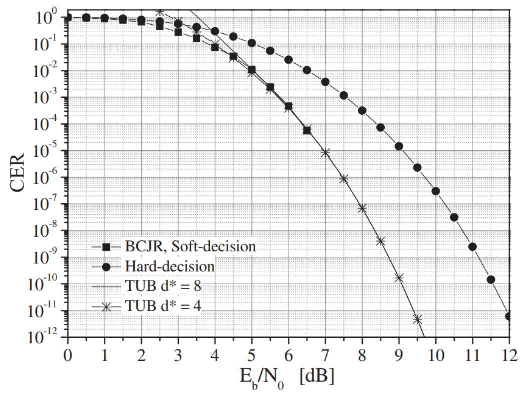


Figure 12*: Performance of the BCH (63, 56) code with hard- and soft-decision decoding*

The hard decision curve is of our interest, the others refer to suggestions in order to improve the performance and can be found in [8].

Finally, an implementation and its results can be found in [9] (with MATLAB pseudocode for the algorithm's simulation) and also an FPGA implementation.

5 Convolutional codes

5.1 General

Convolutional codes are error-correcting codes that generate parity bits via a sliding window to calculate $p > 1$ parity bits by combining various subsets of bits in the window. By using them we send **only** the parity bits. In contrast to classic block codes, the windows overlap and slide by one. The window's size (in bits) is called the code's

constraint length (K). There is the following trade-off when we select how many bits we will have in every window:

By having a larger constraint length

- we can achieve a better resilience to bit errors, but
- it will take longer to decode

Convolutional codes are often characterized by the base code rate and the constraint length of the encoder **[n,k,K]**. The base code rate is typically given as $\frac{n}{k}$, where **n** is the input data rate and **k** is the output symbol rate.

If a convolutional code produces p parity bits per window and the window slides by one bit, its rate will be $\frac{1}{p}$. As the number of parity bits increases we have higher resilience to bit errors. The trade-off is that higher amount of communication bandwidth is devoted to coding overhead. In reality, we want to choose the p and the constraint length to be as small as possible while providing a low enough bit error probability.

5.2 Parity Equations

Figure 13 shows an example of parity equations, which produce parity bits from the sequence of data bits (x).

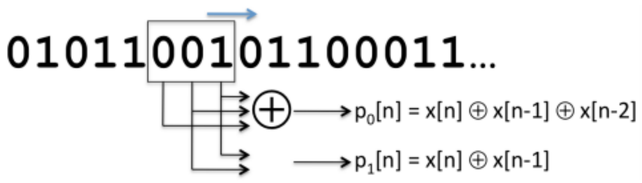


***Figure 13**: Convolutional code with two parity bits per message bit ($p = 2$) and constraint length ($K = 3$)*

In this example, the equations are:

$$p_0[n] = x[n] + x[n-1] + x[n-2] \tag{3}$$

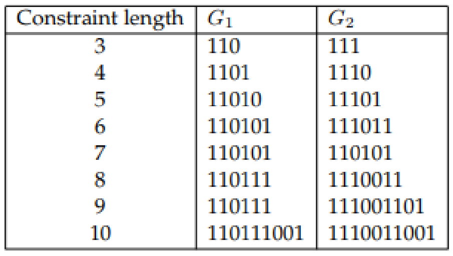
$$p_1[n] = x[n] + x[n-1] \tag{4}$$

In general, one can view each parity equation as being produced by composing the data bits (x) and a generator polynomial, g. In the first example above, the generator polynomial coefficients are (1, 1, 1) and (1, 1, 0).

By g_i we mean the K-element generator polynomial for parity bit p_i. We can then write p_i as follows:

$$p_i[n] = (\sum_{j=1}^{K-1} g_i[j] \cdot x[n-j]) \mod 2 \tag{5}$$

So now it's obvious why we call them convolutional.



Constraint length	G_1	G_2
3	110	111
4	1101	1110
5	11010	11101
6	110101	111011
7	110101	110101
8	110111	1110011
9	110111	111001101
10	110111001	1110011001

***Figure 14**: Examples of generator polynomials for rate $\frac{1}{2}$ convolutional codes with different constraint lengths.*

5.3 Encoding

In [10] you can find the presentation of two views of the convolutional encoder, which are very useful for understanding convolutional codes, encoding and decoding implementation procedures.

To convolutionally encode data, we need **k** memory registers, each holding one input bit. Usually all memory registers start with a value of 0. The encoder has **n** $\bmod 2$ adders (each one of them can be implemented with an XOR gate), and **n** generator polynomials — one for each adder (figure 15). An input bit m_1 is fed into the leftmost register. Using the generator polynomials and the existing values in the remaining registers, the encoder outputs **n** symbols. Now bit shift all register values to the right (m_1 moves to m_0, m_0 moves to m_{-1}) and wait for the next input bit. If there are no remaining input bits, the encoder continues shifting until all registers have returned to the zero state (flush bit termination). The generator polynomials are $G_1 = (1, 1, 1)$, $G_2 = (0, 1, 1)$ and $G_3 = (1, 0, 1)$, so output bits are calculated from the equations below:

$$n_1 = m_1 \oplus m_0 \oplus m_{-1}$$

$$n_2 = m_0 \oplus m_{-1}$$

$$n_3 = m_1 \oplus m_{-1}$$

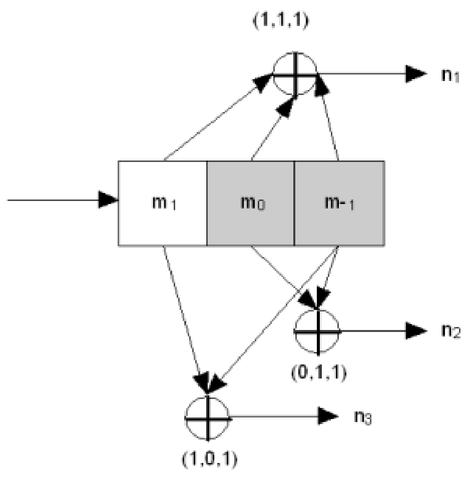


***Figure 15**: Rate $\frac{1}{3}$ non-recursive, non-systematic convolutional encoder with constraint length $K = 3$*

5.4 Decoding problem

A decoder that is able to infer the most likely sequence is called a **maximum likelihood decoder** and it maximizes $\Pr(r \mid c)$, which means that it finds c so that the probability that r was received, given that c was sent, is maximized. If the Hamming distance between two codewords r, $\overline{c}$ equals d and the length of the received codeword is N, then $\Pr(r \mid c) = p^d \cdot (1-p)^{N-d}$ (considering the binary symmetric channel, where bits are received with probability $p < \frac{1}{2}$). By taking the logarithm we have the following equation:

$$log \Pr(r \mid c) = d \cdot logp + (N-d) \cdot log(1-p) = d \cdot log(\frac{p}{1-p}) + N \cdot log(1-p) \qquad (6)$$

We have $p < \frac{1}{2}$, so $\frac{p}{1-p} < 1$ and $logp < 0$. Thus, if we want to minimize $log \Pr(r \mid c)$ we have to minimize d. As a result, the most likely sequence of parity bits that was transmitted must be the one with the smallest Hamming distance from the sequence of parity bits received.

5.5 Decoding convolutional codes

Several algorithms exist for decoding convolutional codes. For relatively small values of k, the Viterbi algorithm is universally used as it provides maximum likelihood performance and is highly parallelizable. Viterbi decoders are thus easy to implement in VLSI hardware and in software on CPUs with SIMD instruction sets.

Longer constraint length codes are more practically decoded with any of several sequential decoding algorithms, of which the Fano algorithm is the best known. Unlike Viterbi decoding, sequential decoding is not maximum likelihood but its complexity increases only slightly with constraint length, allowing the use of strong, long-constraint-length codes. Such codes were used in the Pioneer program of the early 1970s to Jupiter and Saturn, but gave way to shorter, Viterbi-decoded codes, usually concatenated with large Reed–Solomon error correction codes that steepen the overall bit-error-rate curve and produce extremely low residual undetected error rates.

Both Viterbi and sequential decoding algorithms return hard decisions: the bits that form the most likely codeword. An approximate confidence measure can be added to each bit by use of the Soft output Viterbi algorithm. Maximum a posteriori (MAP) soft decisions for each bit can be obtained by use of the BCJR algorithm.

5.6 Different convolutional codes performance

In [1] as we mentioned in 2.1 there is a link level simulation with which we can have an approximation of the function of channel coding. Figure 16 depicts the BER predictions for $R = \frac{1}{3}$, $K = 3$ convolutional code. For the simulation $R = \frac{1}{3}$ and $d_{min} = 8$ block code with codeword length $n_{max} = 46$ or $n_{min} = 24$ is used (we care about the simulations of the figure). The minimum resolvable BER is $2 \cdot 10^{-5}$.

A similar comparison was performed for $R = \frac{3}{4}$, $K = 7$ convolutional code.

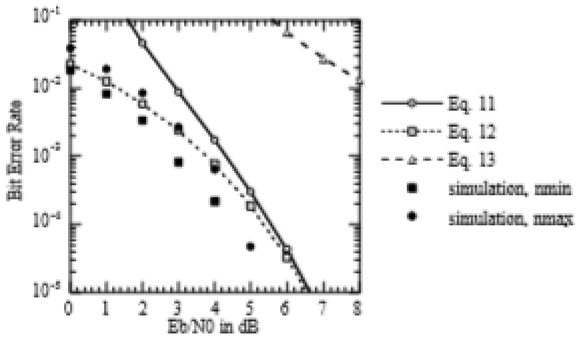


***Figure 16**: BER predictions for a $R = \frac{1}{3}$, $K = 3$ convolutional code*

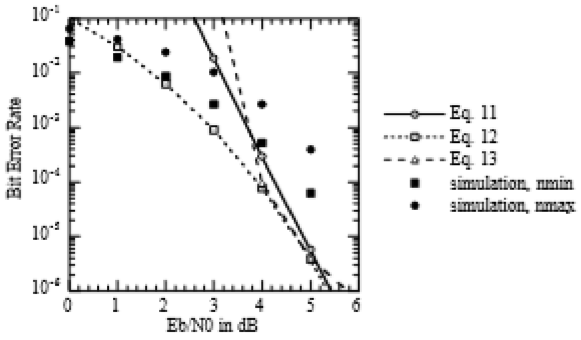


***Figure 17**: BER predictions for a $R = \frac{3}{4}$, $K = 7$ convolutional code*

By returning to [5], which we mentioned in 3.3, although it is considered that the frequency range uplink is 136-146 MHz we can use the results for our purpose. QPSK/OQPSK has been previously selected and the comparison is between coded and uncoded QPSK / OQPSK in Rician & Log-normal channel. For primary simulations Rician and Lognormal fading are used. A code rate that is equal to $R = \frac{3}{4}$ is chosen. For convolutional codes octal generators (171, 133) with constraint length 7 and with vector [110101] as the puncturing vector is used[3]. We should mention that by having higher constraint length the BER becomes better (about 0.5dB per constrain length increase by 1). The negative impact is an increased complexity. The results depict that convolutional codes with $R = \frac{3}{4}$ do not achieve a sufficient BER for Uplink. For a sufficient uplink BER smaller code rates are required. Also, in light shadowing the RS(255,243) code outperforms the other codes as we can see in figures 5 and 6.

In [5] the combination of rate $\frac{2}{3}$ convolutional or RS(255,243) codes with OQPSK is proposed for the uplink in light fading channel conditions. Convolutional codes with code rate $\frac{1}{2}$ are proposed for the strong fading channel. However, we observe that a convolutional code with $R = \frac{1}{3}$ achieves lower BER for lower SNR values. For example, for $SNR = 9dB$ we can have $BER = 10^{-6}$.

[3]Puncturing is a technique used to make a $\frac{m}{n}$ rate code from a "basic" low-rate (e.g., $\frac{1}{n}$) code. It is reached by deletion of some bits in the encoder output. Bits are deleted according to a puncturing matrix.

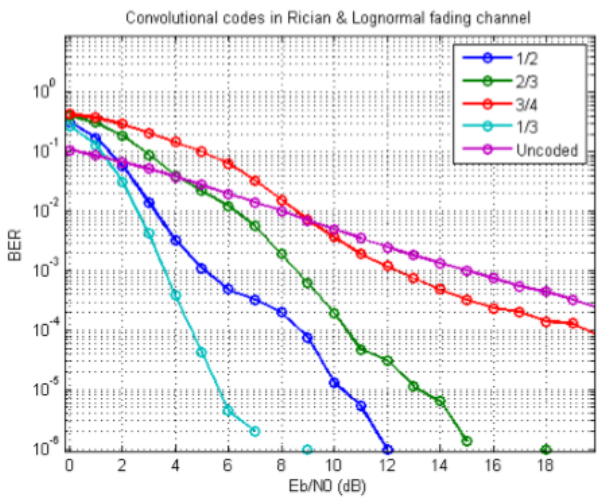


***Figure 18**: Comparison of coded and uncoded QPSK/OQPSK for different code rates in Rician & Log-normal channel (light shadowing $K = 4$, $\mu = 0.13$ and $\sigma = 1.0$*

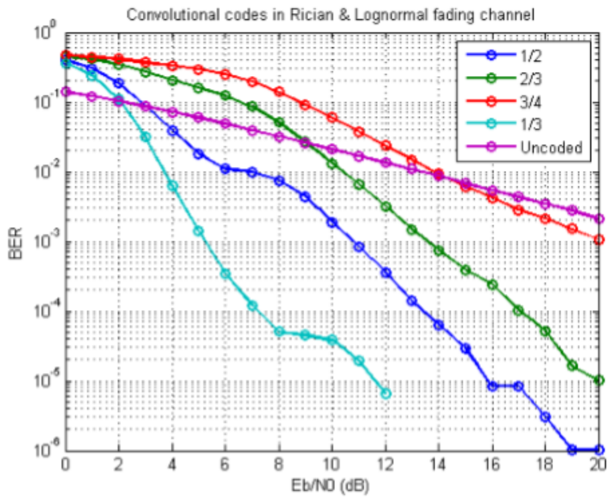
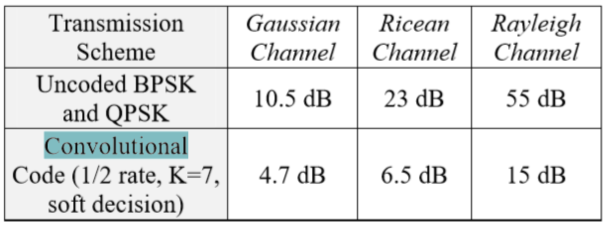


***Figure 19**: Comparison of coded and uncoded QPSK/OQPSK for different code rates in Rician & Log-normal channel (strong shadowing $K = 0.6$, $\mu = -1.08$ and $\sigma = 2.5$*

In Figure 7 (presented in 3.4) we see that Reed Solomon code and convolutional code (the rate is not given but that maybe means that we have the 'mother' rate $\frac{1}{2}$) perform better than the Golay and Hamming codes. It is clear that a convolutional code provides the highest gain[4]. We observe that with convolutional coding we can achieve low BER with lower SNR values comparing them to the others. On the other hand, convolution code requires high power consumption due to its encoding and decoding complexity. Below there are extra cases for convolutional codes:

[4]For those who don't remember it gain is the difference in required SNR to obtain a certain BER for a specific code compared to uncoded system (to obtain higher gain a long code must be used, this demands more complex decoder and high energy consumption).



Transmission Scheme	*Gaussian Channel*	*Ricean Channel*	*Rayleigh Channel*
Uncoded BPSK and QPSK	10.5 dB	23 dB	55 dB
Convolutional Code (1/2 rate, K=7, soft decision)	4.7 dB	6.5 dB	15 dB

Figure 20: *Required SNR for* $BER = 10^{-6}$

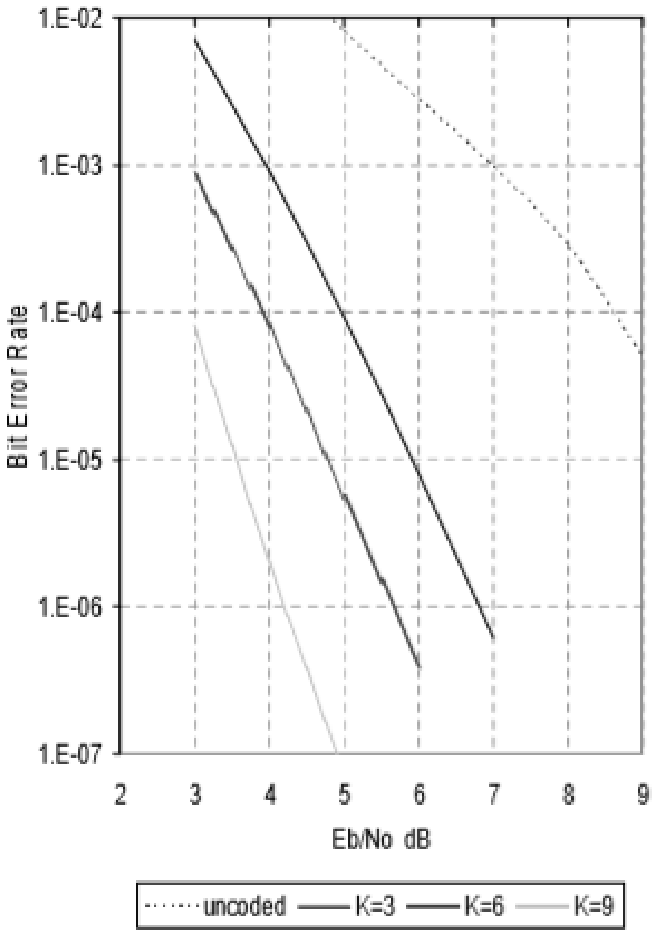


Figure 21: *BER vs.* $\frac{E_b}{N_0}$ *for* $\frac{1}{2}$*-rate convolutional codes (Gaussian channel, BPSK, soft decision)*

6 Conclusion

6.1 Comparison

At last, we can compare specific codes on certain important characteristics, in order to have an idea on which of them suits us better. Firstly, Golay (24,12,8) seems to have the easiest implementation of them all (it can be actuated in hardware with even the smallest micro-controllers). However, the rate of this code is 0.5 and the SNR has to be approximately around 9 dB in order to have BER$\approx 10^{-6}$, (with MSK modulation). On the contrary, RS (511,480,32) has 0.939 rate of code, can correct more bits, although

Golay can increase the number of bits it corrects with interleaving, and is -in terms of BER-SNR relation and gain- better than the previous one, BER $\approx 10^{-6}$ for SNR $\approx 7.6dB$. We should not forget, though, that the codeword length is much bigger in the last case (it can be easily modified as mentioned in 3.1). BCH (63,56) seems to have slightly better BER-SNR curve for $SNR < 8dB$, but its implementation is not as easy as Golay's, it corrects **only** 1 bit in error-correction mode and it has BER $\approx 10^{-6}$ for SNR $\approx 9.5dB$. Last but not least, there are the convolutional codes which are an interesting option. Obviously their implementation is more complex than Golay's but we can see that their SNR required in order to achieve $BER = 10^{-6}$ is much lower than the others'. In QPSK/OQPSK example convolutional codes with rates $R = \frac{1}{3}$, $R = \frac{1}{2}$, $R = \frac{2}{3}$ are much better than the other codes when it comes to BER-SNR relation. The same happens in the MSK example where the soft and hard decision convolutional codes are better than the most of the other codes. But we could say that rates $R = \frac{1}{3}$ and $R = \frac{1}{2}$ are low, so we should prefer $R = \frac{2}{3}$.

6.2 Verdict

RS (511,480,32) seems to be a good solution. By taking into account the size of the uplink data, the easy implementation, the reliability and the analysis in this report (even with the disadvantages of BER-SNR relation and lower rate) I think that we could also use Golay (24,12,8) code (with or without interleaving-this will be determined afterwards). Convolutional codes have lower (but many different) rates than RS, higher implementation complexity than Golay and RS **but** they behave a lot better when it comes to achieving $BER = 10^{-6}$ with low SNR value.

6.3 Interesting case

A good idea might be to use two different error correcting codes in order to be safer, but in this case there must be a further complexity, rate and BER-SNR relation study.

References

[1] Martin Dttling, Simon Saunders, "Bit Error Rate Calculation for Satellite Communication Systems".

[2] Mukta Thankachan, Bhagwat Kakde, Manish Jain, "Design and Simulation of Extended Golay Code on FPGA for Long Distance Applications – A Review", Vol. 7, No. 11, Nov 2016.

[3] Wikipedia, Reed-Solomon

[4] M.C. GENNERO, 0. PAPINI, "UTILIZATION OF ERROR CORRECTING CODES FOR DATA TRANSMISSION SIMULATIONS", Discrete Mathematics 56 (1985) 155-168.

[5] Artur Gaysin, Vladimir Fadeev, Marko Hennhöfer, "Survey of modulation and coding schemes for application in CubeSat systems", The Ministry of Education and Science of Russia No. 8.5635.2017/8.9.

[6] Rajoua Anane, Bouallegue Ridha, Kosai Raoof, "Extensive Simulation Performance Evaluation of Minimum Shift Keying Scheme with Error Correcting Codes in Wireless Sensor Networks", LaboratoryInnov'COM of Higher School of Telecommunication, University of Carthage Tunisia and Laboratory of Acoustics at University of Maine, LAUM UMR CNRS n 6613, France.

[7] Gabriel Marchesan Almeida, Eduardo Augusto Bezerra, Luis Vitório Cargnini, Rubem Dutra Ribeiro Fagundes, Daniel Gomes Mesquita, "A Reed-Solomon Algorithm for FPGA Area Optimization in Space Applications",Second NASA/ESA Conference on Adaptive Hardware and Systems(AHS 2007), IEEE (2007)

[8] M. Baldi, F. Chiaraluce, R. Garello, N. Maturo, I. Aguilar Sanchez, S. Cioni, "Analysis and performance evaluation of new coding options for space telecommand links - Part I: AWGN channels", INTERNATIONAL JOURNAL OF SATELLITE COMMUNICATIONS AND NETWORKING Int. J. Satell. Commun. Network. 2015; 33:509–525 Published online 9 September 2014 in Wiley Online Library (wileyonlinelibrary.com). DOI: 10.1002/sat.1092.

[9] Arunkumar.S, Kalaivani.T, "FPGA Implementation of CCSDS BCH (63, 56) for Satellite Communication", 2012 IEEE International Conference on Electronics Design, Systems and Applications (ICEDSA).

[10] LECTURE 8:Convolutional Coding, MIT 6.02 DRAFT Lecture Notes, October 4, 2010)

[11] Wikipedia, BCH

[12] Wikipedia, Golay

[13] Hank Wallace, Atlantic Quality Design, Inc., "Using The Golay Error Detection And Correction Code", http://aqdi.com/articles/using-the-golay-error-detection-and-correction-code-3/

[14] Wikipedia, Convolutional codes