RS Paper*

*Note: Sub-titles are not captured in Xplore and should not be used

1st Given Name Surname
dept. name of organization (of Aff.)
name of organization (of Aff.)
City, Country
email address

2nd Given Name Surname
dept. name of organization (of Aff.)
name of organization (of Aff.)
City, Country
email address

3rd Given Name Surname
dept. name of organization (of Aff.)
name of organization (of Aff.)
City, Country
email address

4th Given Name Surname
dept. name of organization (of Aff.)
name of organization (of Aff.)
City, Country
email address

5th Given Name Surname
dept. name of organization (of Aff.)
name of organization (of Aff.)
City, Country
email address

6th Given Name Surname
dept. name of organization (of Aff.)
name of organization (of Aff.)
City, Country
email address

***Abstract*—Pending.**

***Index Terms*—pending, pending, pending, pending**

I. Introduction

In modern communications and storage systems, it is observed the increasingly demand for solutions of high speed data rates and reliability with economic viable hardware. Error Correction Coding (ECC) techniques plays an important role to fulfill these opposite requirements. They aim to incorporate redundancy to the transmitted data and restrict the characteristics of the encoded signal in order to augment the decoder's capability to extract the original data from unreliable and noisy communication channels [1]. Reed-Solomon (RS) codes is one of the most popular ECC methods that uses a block-by-block basis to correct burst errors and erasures in data. The adequate compromise between effectiveness and implementation complexity guarantees the widespread use of RS codes in many applications until today.

The paper entitled "Polynomial Codes over Certain Finite Fiels" [2] published in June 1960 was the first formal publication of RS codes. At this time, a pursue to the most optimal and mechanizable ECC emerged after Claude Shannon theory defines the upper bound of the channel capacity at which is possible to transmit error-free information using a proper coding method. Richard Hamming developed the first expressive practical work on ECC algorithms using linear algebra in early 1950s, and then Bose, Ray-Chaudhuri and Hocquenghem generalized his experiments in 1959 - being known as BCH codes. Finally, Irving Reed and Gustave Solomon introduced a special case of BCH codes (RS codes) that employs an efficient decoding algorithm that enabled ~~its~~ wide range use [3].

~~Since its introduction~~, many technologies and standards have adopted RS codes with a variety of parametric configurations. The Compact Disc (CD) system was the first consumer mass application which employed RS codes [4], and since then many other used adopted it such as satellite and mobile systems, DVD and barcodes, and digital television. The literature presents many custom hardware implementations of RS Codecs even though they only differs in parametric terms. Current Hardware Description Languages (HDL) - Verilog and VHDL - already provide ways to express a parametric Register Transfer Level (RTL) architecture; however, it has not been completely explored in previous works. Also, verification of generic RTL blocks might be challenging and requires a proper methodology to assess implementation against block requirements. Most papers that explore RS codecs only provide a limited validation of the RTL implementation without analyzing metrics and results that certifies the block verification.

This work proposes a generic RTL implementation in VHDL and verification of a RS codec. The primary goal of this study is to present an open access IP of a RS codec that could be used as baseline for future works, and not to provide an optimized RTL architecture of it. The block will be verified by using Formal Verification (FV) approach, which is an emerging technology that has been becoming widely adopted in IP designs. Verifying all parametric combinations would not be feasible - or even impossible -, then this paper will use IEEE 802 standard as background. The IEEE 802 group, which develops standards and recommended practices for local, metropolitan, and other area networks, catalogs many commercial RS codec specifications with a diverse range of parametric setups that will be explored in this work.

The paper is organized as in the following: Section II introduces theoretical aspects of RS codes, section III analyzes related works, section IV presents the proposed RTL architecture of the RS codec, section V examines and exhibits results of the adopted verification approach, section VI explores synthesis aspects of the implemented IP and the paper conclusion is presented in section VII.

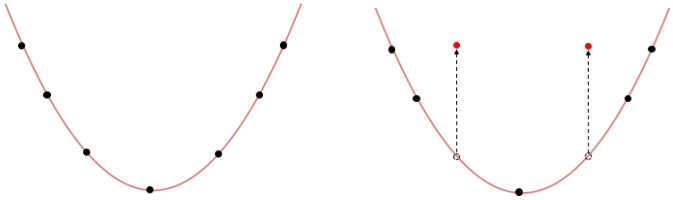
Identify applicable funding agency here. If none, delete this.

II. Reed-Solomon Codes: Theoretical Aspects

What are RS codes? This fundamental question can be the starting point for discussing the theory behind it. The authors of RS codes already answered it using few lines in [5]:

> "They are simply sets of algebraic curves defined by polynomials with a limited range of degrees. These curves when graphed are a set of discrete points - the abscissas and ordinates are values in a Galois Field (GF). The degree limitation allows recovery of the complete curve even when the graph is assumed to be smudged and erased at many points. The relation of this idea to error control on noise-corrupted digital communication channels is immediate."

Even without knowing GF, it is possible to have a glimpse of how RS codes work. It basically relies on "fitting" points in a plane by the polynomial of smallest degree that passes through these points [6]. Basic polynomial mathematics affirms that 2 points can be fitted by a straight line, 3 points by a parabola, and so on. If some redundancy is added by oversampling a given polynomial function, the curve can be correctly guessed even though some recorded points move unintentionally. For instance, assuming that a parabola, which can be represented by $k = 3$ points, is recorded in $n = 7$ points (Fig. 1a). If $t = 2$ points are moved vertically, the original polynomial can still be recovered by finding a single parabola that fits as many as possible of the 7 points (Fig. 1b).



(a) (b)

Fig. 1: Parabola sampled in $n = 7$ points (a). The same parabola is still recovered even though two recorded points are erroneously moved (b).

~~It is known that~~ if t is greater than half of the number of extra points, an incorrect polynomial function may fit the greatest number of points. Therefore, $t = (n-k)/2$ represents the maximum capacity of error correction. ~~RS codes can be seen in such way, but applied in the universe of field arithmetic. Therefore,~~ RS codes are represented as a set of length n vectors (codewords) where their elements (symbols) consist of q binary digits (Fig. 2). The original message takes k vector positions, then there are $n - k$ redundant symbols (parity). A RS codec configuration is usually referred by the notation RS(n, k).

RS codes are linear, all codewords are sums of codewords, and cyclic, every cyclic shift of a code word is also a codeword [8]. These properties make the hardware implementation of the codec feasible, and this is granted only because of the application of GF during encoding and decoding processes.

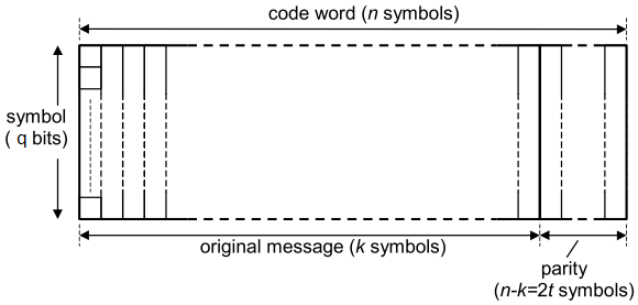


Fig. 2: Generic parametric configuration of RS codes (figure adapted from Fig.1s in [8]).

Its algebra establishes restricted rules in a finite field, thus every arithmetical operation will result in an element within a finite number of elements. GF is built using a primitive element α which is root of a primitive polynomial $(p(x))$ used as reference to generate the field elements. For digital world it is convenient to choose $\alpha = 2$ since it represents the binary notation. Hence, the total number of elements in a GF corresponds to 2^q. Table 1 is a example of GF(2^3) where $\alpha = 2$ is root of $p(x)$. Hence, $\alpha^3 + \alpha + 1 = 0$, or equivalently $\alpha^3 = \alpha + 1$ (additions and subtractions are interchangeable in field arithmetic).

TABLE I
GF(2^3) representation using $p(x) = x^3 + x + 1$

Index Form	Polynomial Form	Decimal Form
0	0	0
α^0	1	1
α^1	α	2
α^2	α^2	4
α^3	$\alpha + 1$	3
α^4	$\alpha^2 + \alpha$	6
α^5	$\alpha^2 + \alpha + 1$	7
α^6	$\alpha^2 + 1$	5

The approach to generate RS codes published in the ~~original~~ paper [2] takes a sequence of symbol information, $\{m_{k-1}, m_{k-2}, ..., m_1, m_0\}$, to build the polynomial $M(x) = m_{k-1}x^{k-1} + m_{k-2}x^{k-2} + ... + m_1x + m_0$. If $n \leq \alpha^m$ elements from GF(α^m) are evaluated in $M(x)$, then the result is a system of n linear equations in k variables (1). This system can be solved using any $\binom{n}{k}$ combinations of equations. If the number of corrupted symbols $t \leq (n-k)/2$, the most occurring solution of $\binom{n}{k}$ linear systems corresponds the original message. However, this method clearly requires much computational resources to calculate all possible solutions.

$$
\begin{aligned}
&M(0)=m_0\\
&M(\alpha)=m_{k-1}\alpha^{k-1}+\ldots+m_2\alpha^2+m_1\alpha+m_0\\
&M(\alpha^2)=m_{k-1}\alpha^{2(k-1)}+\ldots+m_2\alpha^4+m_1\alpha^2+m_0\\
&\ldots\\
&M(\alpha^{2^m-1})=m_{k-1}\alpha^{(2^m-1)(k-1)}+\ldots+m_2\alpha^{(2^m-1)2}+m_1\alpha^{(2^m-1)}+m_0
\end{aligned}
\tag{1}
$$

The most practical approach uses Generator Polynomials (GP) (2) for codification. The roots of GP are composed by consecutive elements of the adopted GF at extent of 0 to t. It can be demonstrated that cyclic codes can always be defined by a GP [3], which implies that the resulting codeword contains coefficients of a polynomial that is divisible by the GF. This assumption culminates in a more efficient implementation for RS codec [9]. The next topics describe encoding and decoding processes of RS codes using GPs.

$$G(x) = (x + \alpha^0)(x + \alpha^1)...(x + \alpha^{2t-1}) \tag{2}$$

A. Encoding Process

The encoding process exploits the polynomial division of $M(x)$ by $G(x)$ in a GF(α^q), as demonstrated by (3). $M(x)$ shifted by $n-k$ positions (multiplication by x^{n-k}) in order to reserve the lowest $n-k$ positions to the parity check symbols. $Q(x)$ and $R(x)$ are the quotient and the remainder of the division operation. If $\{m_{k-1}, m_{k-2}, ..., m_1, m_0\}$ is assigned to be the first k elements of the codeword that forms $C(x)$, then the coefficients of $R(x)$ are the appropriate elements to shape $C(x)$ in such way that it is divisible by $G(x)$. Equation (4) shows the result of the encoding process.

$$\frac{x^{n-k}M(x)}{G(x)} = Q(x)G(x) + R(x) \tag{3}$$

$$\begin{aligned} C(x) &= x^{n-k}M(x) + R(x) \\ &= m_{k-1}x^{n-1} + ... + m_0x^{n-k} \\ &\quad + r_{n-k-1}x^{n-k-1} + ... + r_0 \end{aligned} \tag{4}$$

B. Decoding Process

- General description of the process
- Diagram

1) Syndrome:

- Principle
- Algorithm

2) Euclidean:

- Principle
- Algorithm

3) Chien Search:

- Principle
- Algorithm

4) Forney:

- Principle
- Algorithm

C. RS in IEEE 802

- Introduction and details
- Table with the RS configurations
- Plot BER vs SNR for all configs

III. RS Codes: RTL Design

Add one generic paragraph

A. Encoder

According to II-A, the encoded codeword consists of the ~~own~~ input data symbols and the result of the mod operation between $x^{n-k}M(x)$ and $G(x)$. Besides that, it is required a ~~proper~~ module interface that enables coordination of control operations such as informing starting and ending of codewords, data validity and other aspects. Table 2 describes all RS Encoder ports.

TABLE II: RS Encoder ports

Name	I/O	Description
clk	I	System clock pin
rst	I	System reset pin
i_start_codeword	I	Indication of input codeword start
i_end_codeword	I	Indication of input codeword end
i_valid	I	Validity of input symbols and indicators
i_symbol	I	Input data symbol
o_start_codeword	O	Indication of output codeword start
o_end_codeword	O	Indication of output codeword end
o_in_ready	O	Ready to accept input symbols and indicators
o_valid	O	Validity of output symbols and indicators
o_error	O	Unexpected behavior happened
o_symbol	O	Output data symbol

The top level architecture has control and processing units as seen in Fig. 3. The processing unit is only responsible for computing the mod operation and output the encoded symbols. It is managed by the control unit, which initializes and interrupts the processing unit and outputs reference signals. Also, input symbols are delayed by one cycle as the control unit requires such delay to know which action should be done with the input symbol. There is an output register as well responsible for synchronizing all output signals of the RS Encoder. The only exception is o_in_ready because a readily response is required for this signal, which is a reference to blocks that drives RS Encoder inputs.

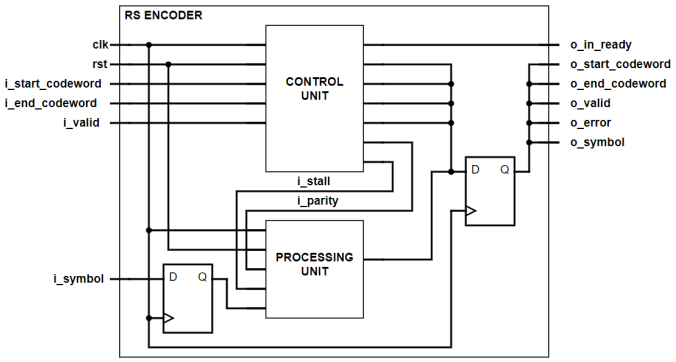


Fig. 3: Top level architecture for RS Encoder.

1) Processing Unit: The mod operation can be performed by a Linear Feedback Shifter Register (LFSR). The remainder is incrementally computed for the first k input symbols, which represents the coefficients of $C(x)$ starting from the highest order. The depth of the LFSR corresponds to the number of parity symbols, which is n - k. Fig. 6 shows the schematic of the Processing Unit. Originally, subtractions are required

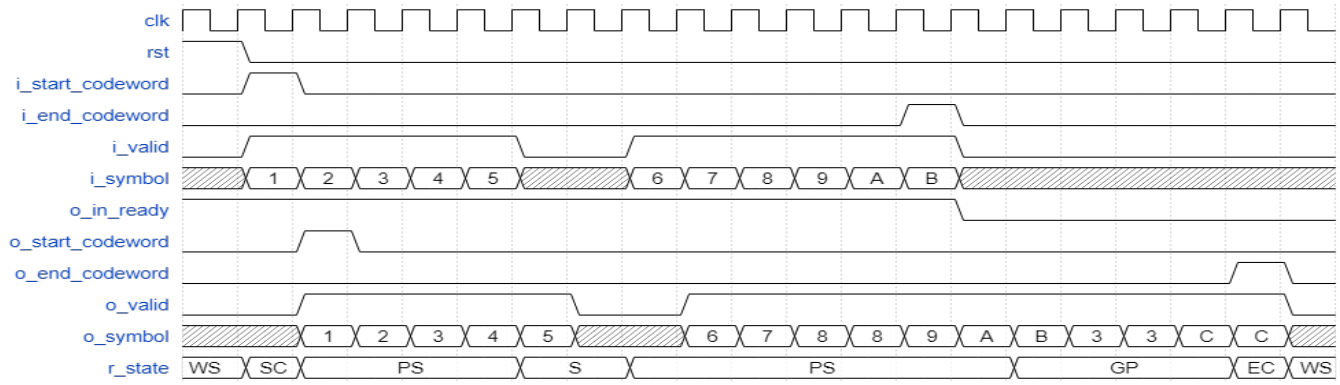


Fig. 4: Waveform example of the encoding process in GF(2^4)

during the calculation of the reminder, but sum is equivalent to subtraction in GF. The GF adder A_div is responsible for finding the quotient term. This number is multiplied by the GP coefficients, with the exception of the highest (x^{2t} according to equation 2) that was already canceled in the current division operation step. The multiplication result is them summed (or subtracted) with the previous terms and stored into the registers. After k steps, the remainder result is completed, and then it can be output by selecting the first input of the output mux (M_out). Every register has a mux in its data input that may conserve its state when the mod calculation is interrupted.

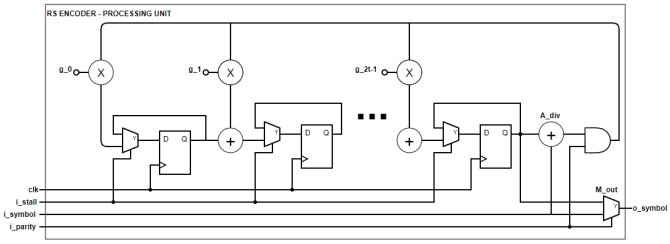


Fig. 5: Design implementation of Processing Unit

2) Control Unit: The Control Unit implements the Finite State Machine (FSM) showed in Fig. 7. WAIT_SYMBOL (WS) is the initial state and looks for the first valid input symbol. Once it is found, the next state is START_CODEWORD (SC), which enables the Processing Unit to receive the sequence symbols to be encoded and also asserts o_start_cw for 1 cycle. Then, PROCESS_SYMBOLS (PS) takes over and supervises the input symbols until the input message ending, signaled by i_end_cw. The transition condition is related to r_cnt, which is a counter that indicates how many symbols were processed. When it reaches the maximum number of allowed input symbols (MAX_I_SYM) having a previous i_end_cw signaling, the FSM is ready to go to GENERATE_PARITY (GP). This state is responsible for controlling the output of parity symbols stored into the registers of the Processing Unit. r_cnt is incremented each cycle until it is equal to NUM_SYM, which means that all parity symbols were outputted. Finally, the Control Unit will return to WS or SC, depending on whether a new message is available or not. STALL and ERROR states are related to interruptions and unexpected behaviors. Whenever i_valid is de-asserted during input message reception (SC and PS), STALL state assumes to freeze the Processing Unit until i_valid is re-enabled. The ERROR state is reached when i_start_cw and i_end_cw assume unexpected values at a given state (refereed by unexpected in Fig. 7). Once ERROR state is reached, it is required to reset the RS Encoder.

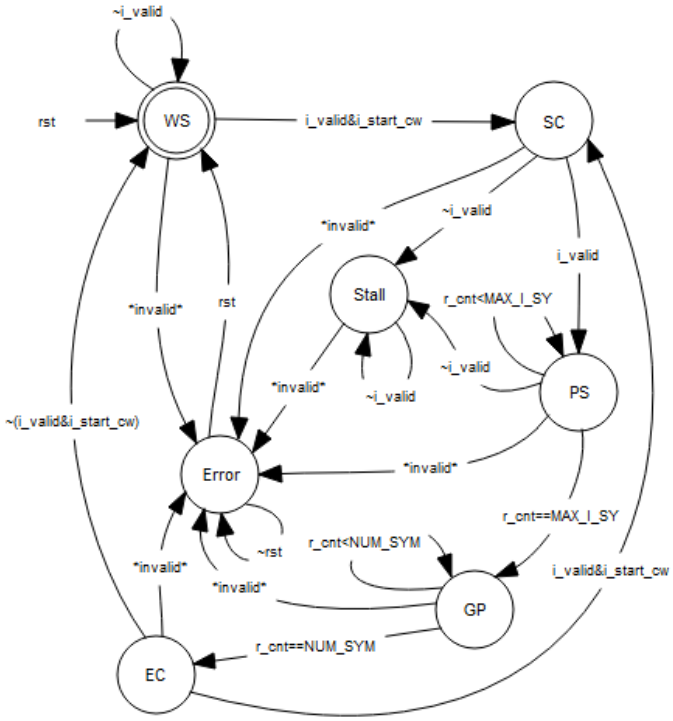


Fig. 6: FSM designed in Control Unit

B. RS Decoder

- Interface and explain how it works. Show waveform example for the most common operator conditions. Use https://wavedrom.com/ for it - no print screens from simulator waveform visualizer.

- Diagram with architecture and explanation

1) Syndrome:

- Architecture
- State Machine

2) Euclidean:

- Architecture
- State Machine

3) Chien Search:

- Architecture
- State Machine

4) Forney:

- Architecture
- State Machine

5) FIFOS:

- Calculation of FIFO size

IV. REED SOLOMON - VERIFICATION

- Explain the challenges of verifying a generic RTL. Introduce formal verification and explain why it fits this problem.
- Methodology. Use this reference: A Coverage-Driven Formal Methodology for Verification Sign-off

A. RS Encoder

- Testplan
- Formal test bench arch.
- Results

B. RS Decoder

- Testplan
- Formal test bench arch.
- Results

C. Verification bug tracking

Show plots about number of bugs found and time.

D. Formal effort analysis

The idea here is show how formal verification scales (or not) with more complex RS configurations

V. REED SOLOMON - SYNTHESIS

- Estimation of Area and number of gates. Here we can compare our IP with commercial IPs.
- Synthesis report
- Critical path and maximum clock

VI. REED SOLOMON - IMPLEMENTATION DETAILS

- show the vhdl file structure and tests
- Explain pyhton script for generating constants

VII. REED SOLOMON - CONCLUSION

...

ACKNOWLEDGMENT

The preferred spelling of the word "acknowledgment" in America is without an "e" after the "g". Avoid the stilted expression "one of us (R. B. G.) thanks ...". Instead, try "R. B. G. thanks...". Put sponsor acknowledgments in the unnumbered footnote on the first page.

REFERENCES

Please number citations consecutively within brackets [1]. The sentence punctuation follows the bracket [2]. Refer simply to the reference number, as in [3]—do not use "Ref. [3]" or "reference [3]" except at the beginning of a sentence: "Reference [3] was the first ..."

Number footnotes separately in superscripts. Place the actual footnote at the bottom of the column in which it was cited. Do not put footnotes in the abstract or reference list. Use letters for table footnotes.

Unless there are six authors or more give all authors' names; do not use "et al.". Papers that have not been published, even if they have been submitted for publication, should be cited as "unpublished" [4]. Papers that have been accepted for publication should be cited as "in press" [5]. Capitalize only the first word in a paper title, except for proper nouns and element symbols.

For papers published in translation journals, please give the English citation first, followed by the original foreign-language citation [6].

REFERENCES

[1] Geisel, William A. "Tutorial on Reed-Solomon error correction coding." (1990)

[2] Reed, Irving S., and Gustave Solomon. "Polynomial codes over certain finite fields." Journal of the society for industrial and applied mathematics 8.2 (1960): 300-304.

[3] Stephen B. Wicker; Vijay K. Bhargava., (1994) "An Introduction to Reed-Solomon Codes," Prentice-Hall

[4] KAS Immink, "Reed-Solomon Codes and the Compact Disc" in SB Wicker and VK Bhargava, Eds., Reed-Solomon Codes and Their Applications, IEEE Press, 1994.

[5] Reed, Irving S., and Gustave Solomon. "Reed-Solomon Codes: A Historical Overview" in SB Wicker and VK Bhargava, Eds., Reed-Solomon Codes and Their Applications, IEEE Press, 1994.

[6] Jack Keil Wolf, "An Introduction to Reed-Solomon Codes", www.ece.tamu.edu/ hpfister/courses/ecen604/rspoly.pdf.

[7] M. Young, The Technical Writer's Handbook. Mill Valley, CA: University Science, 1989.

[8] Clarke, C. K. P. "Reed-Solomon error correction." BBC R&D White Paper, WHP 31 (2002).

[9] See link below