International Journal of Advances in Engineering Research http://www.ijaer.com

(IJAER) 2014, Vol. No. 7, Issue No. II, February **e-ISSN: 2231-5152, p-ISSN: 2454-1796**

VHDL IMPLEMENTATION OF REED SOLOMON CODING

Mohan K. V.

Lecturer in Electronics Engg.
Govt. Polytechnic College, Kannur, Kerala.

ABSTRACT:

Reed-Solomon codes are very effective at correcting random symbol and random burst errors, and they are widely used for error control in communication and data storage systems ranging from deep-space telecommunications to compact disc. Concatenation of these codes as outer codes with simple binary codes as inner codes provides high data reliability while reducing decoding complexity. RS codes are non-binary cyclic error correcting block codes. Error reduction is critical in today's communication systems, which include wireless, satellite, and space communication. Because data may become corrupted during message transfer, the wireless communication system's high bit error rate necessitates the use of various coding methods for data transfer. Channel coding for error detection and correction aids communication system designers in reducing noise during transmission. The main components of this paper are the implementation of reed Solomon error detecting and correcting code using VHDL language and the Berlekamp-Massey Algorithm. This operation makes use of a Field Programmable Gate Array (FPGA).

Keywords: VHDL, Reed-Solomon, Coding, RS codes, Channel coding, (VHSIC Hardware description Language), Berlekamp Massey, Reed-Solomon Encoder.

INTRODUCTION:

Reed-Solomon codes are error-correcting codes that are block-based and have many applications in digital communications and storage.

- Storage device (hard disks, compact disks, DVD).
- Wireless communication (mobile phone, microwave links).
- Digital television.
- Satellite communication.
- Broadband modems (ADSL).

Reed Solomon codes function by adding "redundant bits" to the original data. The encoded data can then be sent or saved. During transmission, errors can occur for a variety of reasons, such as scratches on the CD, radio frequency interference with mobile phone reception, noise, and so on. The decoder at the receiving end detects and corrects a predetermined number of errors that occurred during transmission. [1]

RS codes are a subset of BCH codes and are linear block codes. An RS code is built on finite fields, also known as Galois fields. The number and type of errors that can be corrected are determined by the Reed-Solomon code's characteristics. The parameters of the RS (n, k) codes are as follows. [2]

1. m - Number of bits per symbol
2. k- Uncoded message length in symbols
3. n- Codeword length in symbols
4. (n-k)- Number of parity check symbols
5. t -Number of correctable symbol errors and 2t = =n-k

For broadcasting, many digital signalling systems employ Forward error correction. It is a technique in which redundant information is added to the signal to allow the receiver to detect and correct transmission errors. For this purpose, many different types of codes have been developed, but Reed-Solomon codes have proven to be a good compromise between efficiency and complexity. Reed-Solomon codes are used in a wide range of applications, including satellite communication, digital audio tape, and CD ROMs. Many different RS codes are required for such device applications. The designer of modern digital systems is then faced with the problem of implementing these various error-correction codes. The use of FPGAs or gate arrays, as well as a hardware description language such as VHDL, has greatly reduced the difficulty of this task. The approach is to write the system's functionality top-down using VDHL, then map the code into hardware elements via the synthesis process, and finally simulate the resting net list using a suitable test bench. [3]

REED Solomon (RS) codes [4], encoders, and decoders are extremely powerful error correction tools that significantly improve transmission quality. RS codes process data by dividing the message stream into data blocks and adding redundancy per block based solely on the current inputs. The symbols in RS coding are finite field or Galois Field (GF) elements. A GF is a set with a finite number of elements. The remainder of a GF polynomial division is appended to the message to encode it. A Linear Feedback Shift Register (LFSR) implementation is used to accomplish this division. The RS encoding mathematics is based on finite field arithmetic [5], [6]. The information block is encoded using GF multipliers. The decoder's function is to process the received codeword and estimate the original message symbols. Decoding and correcting corrupted data typically requires ten times more resources. The codes have the format RS (n, k), where n represents the total number of s-bit wide symbols and k represents the number of s-bit wide information (data) symbols in a codeword. The RS Decoder detects and corrects information (data) symbols in a codeword. The RS encoded data is examined to see if any errors occurred during transmission. After determining the number of errors, the decoder determines whether they are within the range of correction. Following this determination, the decoder corrects the errors in the received data.

In general, RS codes are represented as an RS (n, k) with m-bit symbols, where

Block Length : n

No. of Original Message symbols : k

Number of Parity Digits : n - k = 2t

Minimum Distance : d = 2t + 1.

Figure 1 depicts a codeword based on these parameters diagrammatically:

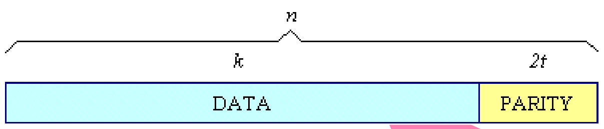

Fig. 1 The Structure of a RS Codeword [7]

Such a code can correct up to (n-k)/2 or t symbols (each symbol is an element e of GF ($2^{(m)}$) (i.e., any corrupted t symbols (single- or multiple-bit errors) can still lead to message recovery). RS codes have the highest code rate of all binary codes and are one of the most powerful burst - error correcting codes. They are especially good at dealing with burst errors because, even if a symbol has all of its bits in error, this counts as only one symbol error in terms of the code's correction capacity.

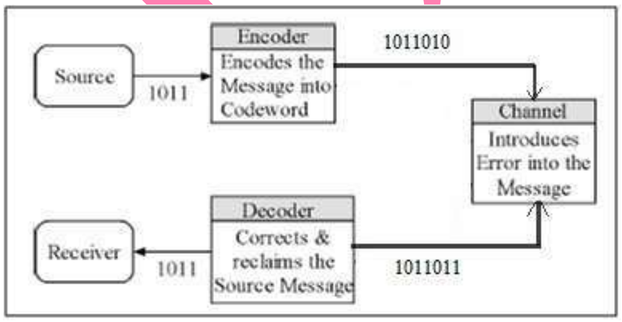
Source
1011
Encoder
Encodes the Message into Codeword
1011010
Channel
Introduces Error into the Message
1011011
Decoder
Corrects & reclaims the Source Message
1011
Receiver

Figure 2: Reed Solomon Protected Channel [8]

This paper describes an efficient RS Encoder and Decoder design and VHDL implementation. To present the paper, the following Chronology is being used.

RS ENCODER

The Reed-Solomon encoder takes k data symbols, computes n - k parity symbols, and appends the parity symbols to the k data symbols for a total of n symbols. The encoder is essentially a 2t tap shift register with m-bit registers. The multiplier coefficients are the RS generator polynomial coefficients. The general idea is to build a polynomial; the coefficients produced will be symbols such that the generator polynomial divides the data/parity polynomial exactly [9].

As shown below, the transmitted codeword is systematically encoded and defined in terms of the transmitted message m(x), the generator polynomial g(x), and the number of parity symbols 2t.

c(x)=m(x) * 2t + m(x)modg(x)

Where, g(x) is the generator polynomial of degree 2t and given by,

$$g(x) = \prod_{i=m_0}^{m_0+2t-1} (x + \alpha^i)$$

OPERATION

Because RS codes are systematic, the information symbols in the codeword are encoded as higher power coefficients. This necessitates shifting information symbols from power level (n-1) to (n-k) and filling the remaining positions from power (n-k-1) to 0 with zeros. As a result, any RS encoder design should be capable of performing the following two operations: division and shifting. Both operations are simple to implement with Linear-Feedback Shift Registers [9][10].

Reed-Solomon codes can be shortened (conceptually) by making a number of data symbols zero at the encoder, not transmitting them, and then inserting them again at the decoder. The encoder is essentially a 2t tap shift register with m-bit registers. The multiplier coefficients are the RS generator polynomial coefficients. The general idea is to build a polynomial; the coefficients produced will be symbols such that the generator polynomial divides the data/parity polynomial exactly. Figure 3 depicts an encoder block diagram.

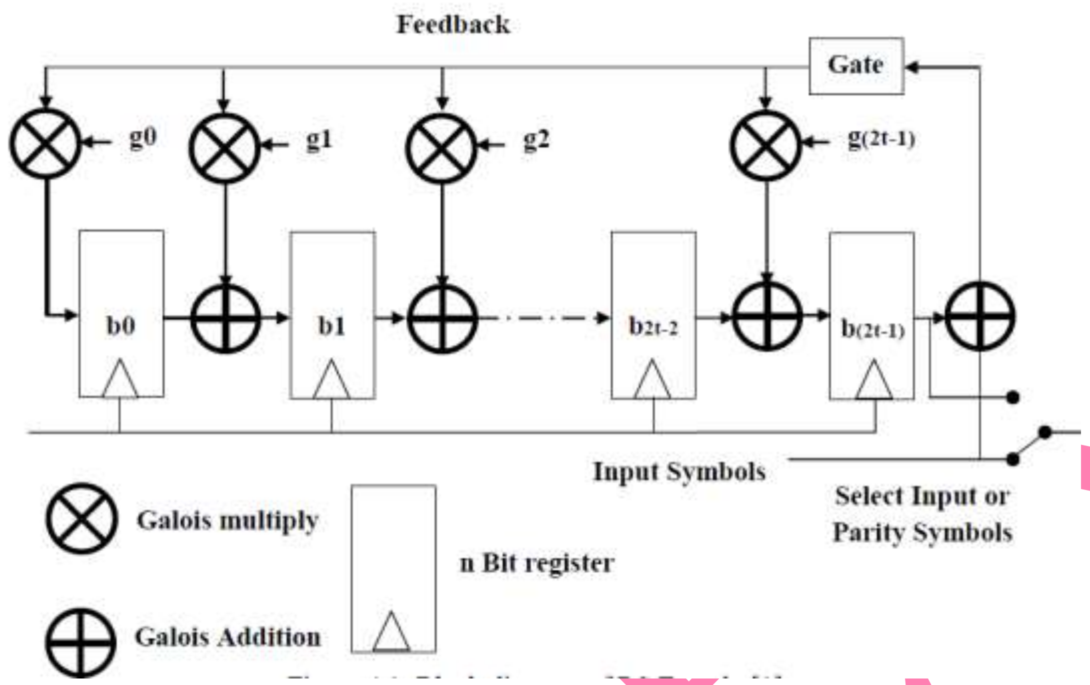

Figure 3: Block diagram of RS Encoder [11]

According to the encoder block diagram, one input to each multiplier is a constant field element that is a coefficient of the polynomial g(x). The information polynomial M(x) is fed into the encoder symbol by symbol for each block. After a predetermined latency, these symbols appear at the encoder's output, where control logic feeds them back through an adder to produce the associated parity. This process is repeated until all k symbols of M(x) are fed into the encoder. [11]

PROPOSED WORK:

The extended inversion less Massey-Berlekamp algorithm is used in this paper. By computing both the error - locator polynomial and the error - evaluator polynomial at the same time, the extended inversion less Massey-Berlekamp algorithm overcomes the extra latency. (255,239) reed Solomon codes have been designed in this thesis. A pipelined RS decoder is proposed in order to reduce hardware complexity and improve clock frequency in RS decoders. This design also employs a pipelined GF multiplier in the syndrome computation block, KES block, Forney block, Chien search block, and error correction block for increased clock frequency, and it has a lower complexity than conventional ME algorithm architectures.

RESULT AND DISCUSSION:

Implementation

Consider the VHDL implementation of the RS encoding. The timing diagram depicts the RS encoder's input/output interface. Is depicted in the figure. The encoding process is initiated by a signal known as input strobe. It is a single clock cycle pulse that comes one clock cycle before the message data called data in. The data consists of the message polynomial's k coefficients. Another requirement for the encoder is the message size: this encoder can handle messages of various sizes. The signal data size communicates the message size. The timing diagram depicts two internal control signals, namely do calc and count. These will be covered in greater detail in the following section. Finally, there are two output signals, output strobe and data out, whose names are self-explanatory. output data.

The Modelsim simulation of the VHDL code above using the message generated by the above Matlab script is shown in Figure 4 below. [12]

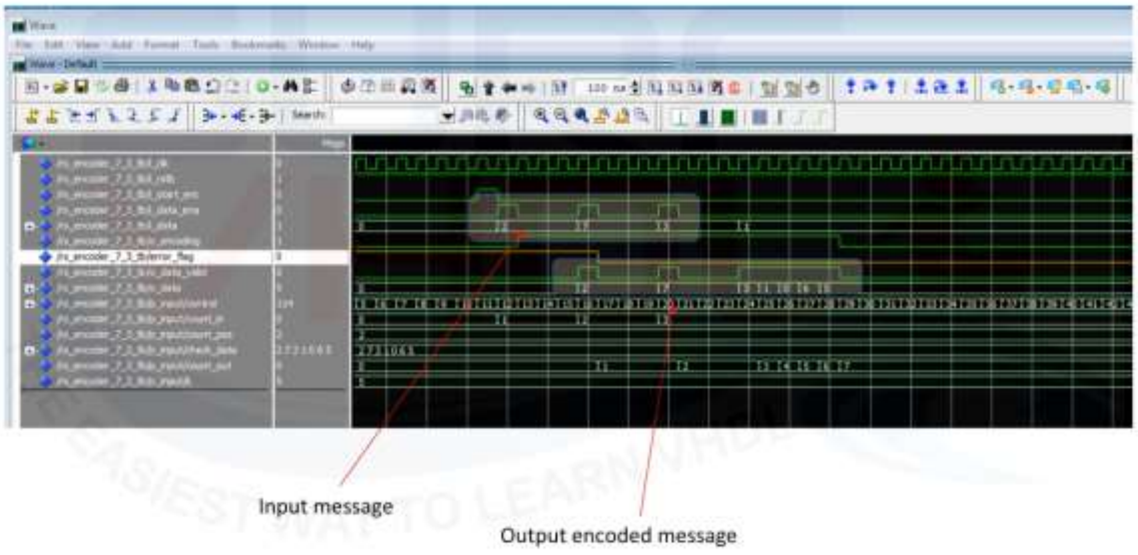

Figure 4: RS encoder VHDL simulation with toggling enable

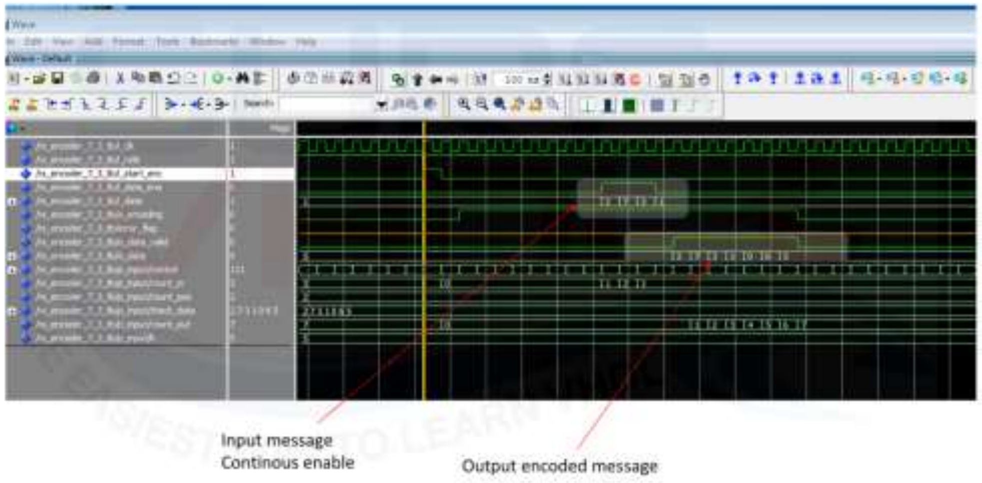

Figure 5: RS encoder VHDL simulation input continuous playing enable

The yellow signal is used to aid in message matching with Matlab reference encoded messages. [13]

VHDL-7-5-Reed-Solomon:

VHDL implementation of Reed-Solomon encoding and decoding (7,5) A Reed-Solomon (RS) Code is a cyclic error correction and detection code that transmits and receives messages using symbol patterns rather than standard bit patterns. Each symbol is made up of m different bits, and the correlation between bits and symbols is done using Finite Fields theory.

This code includes an implementation of the Reed-Solomon Encoder and Decoder, as well as a channel simulation. The following schematics represent the entire system: [14]

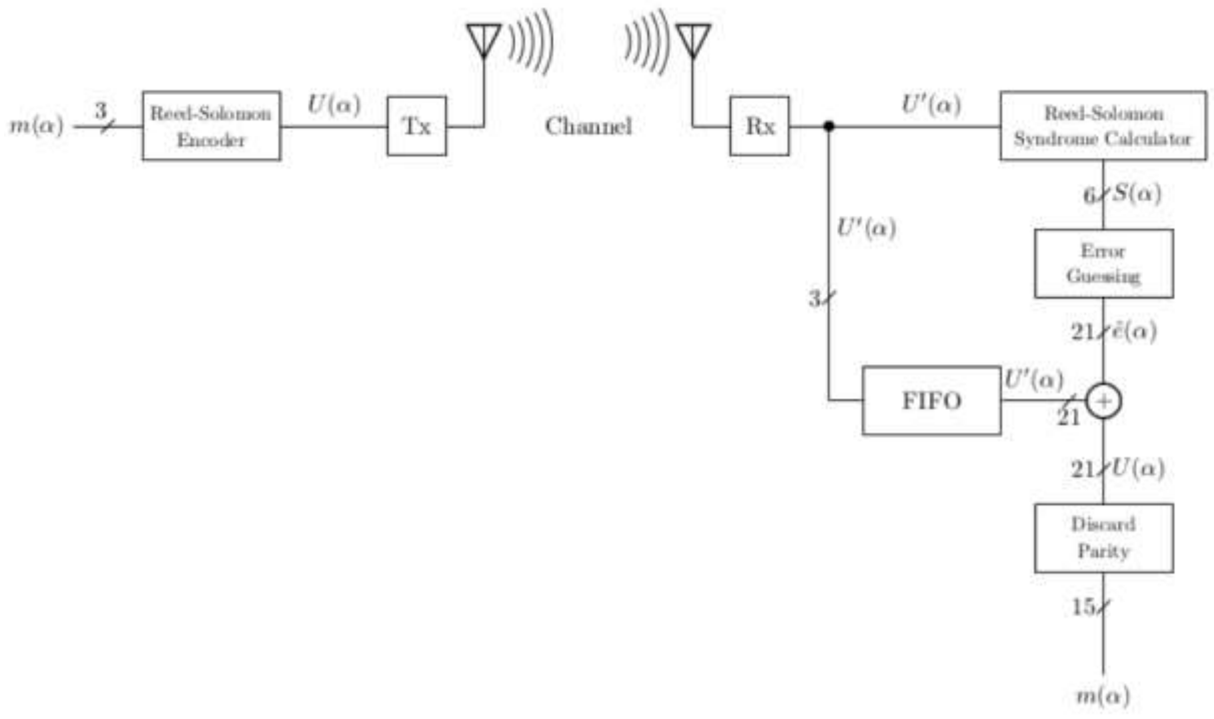

Figure 6: VHDL-7-5-Reed-Solomon

Because two symbols are being multiplied, they enter the system depicted in the following figure. Each symbol should pass through a Symbol Power Encoder after entering the system. This encoder will convert each symbol power to its decimal equivalent.

Following the successful simulation of the written VHDL modules, the (15, 11) and (255,239) Reed-Solomon encoder and decoder were implemented by generating programme files via the QUARTUS II package for each implementation case. These programme files are then transferred to the FPGA ALTERA-FLEX10K10 board, which is depicted in Fig.(9). Internally, the data are applied to this development board by writing a VHDL module that generates such input data. The coded messages are recognised with the help of the board's seven segment display. To recognise the values of coded messages symbols displayed in hexadecimal form, this operation should be performed at a low data rate. [15]

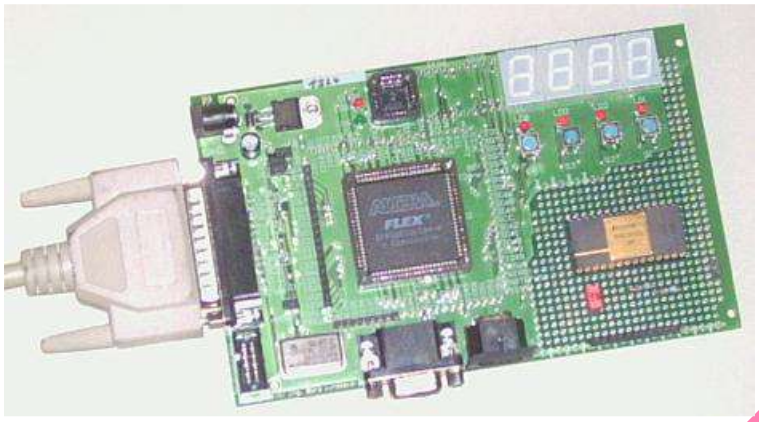

Figure 7: ALTERA-FLEX device family FLEX10K development board

CONCLUSION:

This paper introduced and demonstrated the extended inversion less Massey-Berlekamp algorithm for solving the key equation in a Reed-Solomon decoder, which is a minor improvement over the inversionless Massey-Berlekamp algorithm. It was also presented the error location and correction equations for any arbitrary RS code with an error correcting capability of more than or equal to three errors. Using this information, VHDL-based designs for specific RS codes for both encoder and decoder were implemented. After that, a programme was written to automatically generate the VHDL code for either an RS encoder or an RS decoder based on user inputs.

REFERENCES

1. J.-I. Park, H. Lee (2011) Area-efficient truncated Berlekamp-Massey architecture for Reed-Solomon decoder; 17th February 2011.
2. Mostafa El-Kham et.al. ―Iterative Algebraic Soft-Decision List Decoding of Reed–Solomon Codes‖ IEEE Vol. 24, No. 3, March 2006.
3. E. Prange, (September 1957); "Cyclic Error-Correcting Codes in Two Symbols," Air Force Cambridge Research Center-TN-57-103, Cambridge, Mass.
4. I.S. Reed and G. Solomon, (1960); Polynomial Codes Over Certain Finite Fields,‖ *SIAM Journal of Applied Mathematics,* vol. 8, pp. 300– 304.
5. R. J. McEliece, (1987); Finite Fields for Computer Scientists and Engineers. Boston, MA: *Kluwer Academic*.
6. S. B. Wicker, (1994); Error Control Systems for Digital Communication and Storage, Englewood Cliffs, N.J.: Prentice-Hall.

7. M. Kaur and V. Sharma, (2010); Study of Forward Error Correction using Reed—Solomon Codes, *International Journal of Electronics Engineering,* vol. 2, pp. 331 – 333.
8. Yo Sup (Joseph) Moon, (August 2011); Introduction to Reed-Solomon Codes, Harvard University Department of Mathematics.
9. Reed-Solomon(RS) Coding Overview,‖ VOCAL Technologies, Ltd., Rev. 2.28n, 2010.
10. J.Y Chang and C. Shung, (December 1994); A high speed Reed-Solomon codec chip using look forward architecture*, IEEE APC CAS'94*, PP. 212-217.
11. Sandeep Kaur (2006); VHDL Impletation of Reed-Solomon code,‖ Thesis, Thapar Institute of Engg.
12. K. C. C. Wai and S. J. Yang, (2006); Field Programmable Gate Array Implementation of Reed-Solomon Code, RS (255, 239)‖, New York.
13. Anindya Sundar Das, Satyajit Das and Jaydeb Bhaumik (January 2013); "Design of RS (255, 251) Encoder and Decoder in FPGA", *International Journal of Soft Computing and Engineering (IJSCE)* ISSN: 2231-2307, Volume-2, Issue-6.
14. P.Ravi Tej, Smt.K.Jhansi Rani (August 2013); VHDL Implementation of Reed Solomon Improved Encoding Algorithm" *International Journal of Research in Computer and Communication Technology*, Vol 2, Issue 8.