FPGA Implementation of a High Speed Reed-Solomon Encoder

Vigil Varghese

Department of Electronics and Communication
PES Institute of Technology, Bangalore, INDIA

vigil@pes.edu

***Abstract*— This paper proposes a high speed Reed-Solomon (7,3) encoder architecture. A novel method of performing Galois field multiplication is incorporated into the design, which attributes to the increase in the speed compared to other designs. Theory surrounding the Reed-Solomon codes and Galois field arithmetic is thoroughly explained before moving into the implementation details. The encoder is designed using Verilog HDL. The design is targeted towards Xilinx series of FPGA's. Static timing analysis and area estimation is performed on the design and the results are compared with other known designs.**

***Keywords*— Reed Solomon, Galois field, LSFR, Static timing analysis, Encoder**

I. INTRODUCTION

Reed-Solomon codes are one of the most widely used error control codes. Their application ranges from Compact disc's to deep space telecommunication systems. They are also used in mobile data transmission systems, high-reliability military communication systems, spread-spectrum systems and high-speed super computers [1].

Reed-Solomon codes are a class of non-binary, cyclic, linear block codes. They have very good burst error correcting capabilities, that is, they are effective for channels that have memory [2].

A non-binary block code consists of a set of fixed length code words in which the elements of the code words are selected from an alphabet of C symbols, denoted by

$$C=\{0, 1, 2, \ldots\ldots, C_{n-2}, C_{n-1}\}$$

usually $C=2^m$, so that m information bits are mapped into one of the C symbols. In block coding, the encoder accepts a k-bit message block and generates an n-bit code word. Thus, code words are produced on a block-by-block basis [3].

In a non-binary code each code word is made up of multiple bits. It is different from that of a binary code, in that, the coder operates on multiple bits rather than individual bits.

A code is said to be cyclic if, for any code word

$$C=\{C_0, C_1, C_2, \ldots\ldots, C_{n-2}, C_{n-1}\}$$

the cyclically shifted word

$$C'=\{C_1, C_2, C_3, \ldots\ldots, C_{n-2}, C_{n-1}, C_0\}$$

is also a code word. A code is said to be linear if the sum of any two code words is also a code word.

One reason for the importance of the Reed-Solomon codes is their good distance properties. Reed-Solomon codes achieve the largest possible code minimum distance for any linear code with the same encoder input and output block lengths. For non-binary codes, the distance between two code words is defined as the number of symbols in which the sequences differ. A second reason for their importance is the existence of efficient hard-decision decoding algorithms, which make it possible to implement relatively long codes in many practical applications where coding is desirable [3].

II. REED-SOLOMON PRELIMINARIES

A Reed-Solomon code having k message symbols and n code symbols is denoted by RS (n, k). There are n-k=2t, parity symbols in this code. 2t is said to be the redundancy in the code word. Greater the redundancy, the greater will be the error correcting capability of the code.

For the most conventional RS code

$$(n, k) = (2^m-1, 2^m-1-2t)$$

where t is the symbol-error correcting capability of the code. The code is capable of correcting any combination of t or fewer errors, where t can be expressed as

$$t = [(n-k)/2]$$

The code minimum distance is given by

$$D_{min} = 2t+1$$

An (n, k) linear block code for which the minimum distance equals 2t+1 is called a maximum distance separable (MDS) code. Therefore, every RS code is a maximum distance separable code [4]. MDS codes have a number of interesting properties that lead to many practical consequences.

III. GALOIS FIELD ARITHMETIC

Reed-Solomon codes can easily be described using finite field arithmetic. Finite fields are also known as Galois field (GF). For any prime number p, there exists a finite field denoted by GF(p) that contains p elements. It is possible to extend GF(p) to a field of p^m elements, called an extension field of GF(p), and denoted by GF(p^m), where m is a nonzero positive integer. Symbols from the extension field GF(2^m) are used in the construction of Reed-Solomon codes.

The binary field GF(2) is a subfield of the extension field GF(2^m). The elements of the field GF(2) are {0,1}. If we consider m to be 3 then the elements of the extension field GF(2^3) are {000,001,…….,111}. It is a non-binary field with each element of the field made up of m (here 3) bits. The elements of the extension field GF(2^3) are used in the construction of RS (7,3) codes.

Each extension field GF(2^m) is described by a primitive polynomial. The primitive polynomial for the field GF (2^3) is given by

$$f(x) = 1+x+x^3$$

Additional symbols in the extension field, excluding 0 and 1 are represented by α. Each non-zero element in GF(2^m) can be represented by a power of α. The elements of the field GF(2^3) can be represented as

$$GF(2^3) = \{ 0, \alpha^0, \alpha^1, \alpha^2, \alpha^3, \alpha^4, \alpha^5, \alpha^6\}$$

where α^0,α^1, α^2, α^3, α^4, α^5, α^6 are the roots of the primitive polynomial f(x). The elements of the field in terms of bits are

$$GF(2^3) = \{000,100,010,001,110,011,111,101\}$$

which correspond to 0, α^0, α^1, α^2, α^3, α^4, α^5, α^6 in that order.

Extension field elements can be generated by a linear feedback shift register corresponding to the primitive polynomial. The LSFR (Linear Feedback Shift Register) for GF(2^3) extension field, represented by the polynomial $f(x)=1+x+x^3$ is given in figure 1.

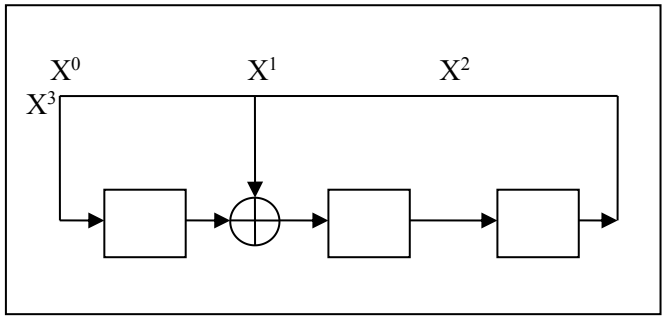

Fig. 1 Extension field generator

Elements of the extension field are generated by shifting the contents of the shift register to right, every clock cycle. Initially the register must contain a non-zero element. It can be observed from figure 1, that there is an XOR (modulo 2 addition) at X^1 and no connection at X^2. The feedback is from X^3 to X^0.

Addition in the extension field is done "modulo 2" between the elements of the field. Multiplication, on the other hand is the "modulo 7" addition of the powers of the roots of the primitive polynomial [5]. For example, $\alpha^2.\alpha^6 = \alpha^8 = \alpha^1$.

IV. Reed-Solomon (7, 3) Encoder

RS (7, 3) codes have n=7 and k=3. They have n-k=2t, i.e., 7-3=4, parity check symbols. They can correct up to t=2 errors.

Let $m=(m_0, m_1, \ldots\ldots\ldots, m_{k-1})$ be a block of k information symbols. These symbols can be associated with an information polynomial

$$m(x) = m_0+m_1x+\ldots\ldots+m_{k-1}x^{k-1}$$

If an (n, k) code is cyclic, it can be shown that the code can always be defined using a generator polynomial

$$g(x)=g_0+g_1x+g_2x^2+\ldots\ldots\ldots+g_{n-k}x^{n-k}.$$

The degree of the generator polynomial is equal to the number of parity symbols. For a (7,3) code there are n-k=4 parity symbols, hence the generator polynomial is given by

$$g(x) = (x-\alpha)\ (x-\alpha^2)\ (x-\alpha^3)\ (x-\alpha^4)$$

where α, α^2, α^3, α^4 are the four roots of the generator polynomial. Solving the above expression we get

$$g(x)= \alpha^3 + \alpha^1x + \alpha^0x^2 + \alpha^3x^3 + x^4$$

If C is a code vector, then it can be represented by n elements as

$$C = \{C_0, C_1, C_2, C_3,\ldots\ldots\ldots\ldots, C_{n-1}\}$$

If the code word is interpreted as a code polynomial then it can be written as

$$C(x) = C_0 + C_1x + C_2x^2 +\ldots\ldots\ldots\ldots+C_{n-1}x^{n-1}$$

A vector C is a code word in the code defined by g(x) if and only if its corresponding code polynomial C(x) is a multiple of g(x). Therefore the encoding procedure can be represented as

$$C(x) = m(x).g(x)$$

V. RS (7, 3) Encoder Implementation

The Reed-Solomon encoder was designed using Verilog HDL. One of the reasons for having higher speed is the use of asynchronous reset instead of synchronous reset in the design. Another reason is the use of a simplified Galois field multiplier, which is the most critical part of the encoder.

A. Encoder Architecture

The encoder is made up of 4 modules as shown in figure 2.

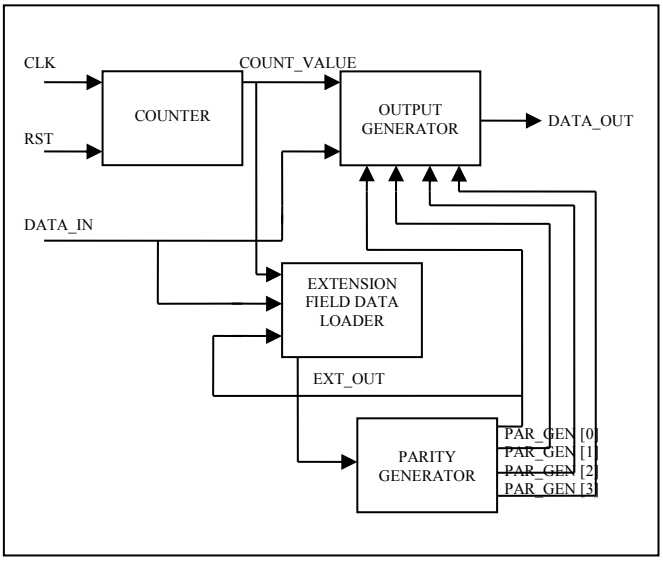

Fig. 2 RS (7, 3) Encoder Architecture

The first module "Counter" increments a variable *COUNT_VALUE* during every positive edge of the clock, if reset (*RST*) is 0 (negative reset). If reset is 1, or *COUNT_VALUE* is 7, then *COUNT_VALUE* and all associated registers are made 0.

The "Output Generator" generates the output in the following manner: If *COUNT_VALUE* is less than 3, then output (*DATA_OUT*) is same as the input (*DATA_IN*). If *COUNT_VALUE* is greater than 3, output is obtained from the parity generator.

The next stage, which is the "Extension Field Data Loader", is the one that aids in the Galois field multiplication. This stage loads the data into the extension field generator (Fig. 1).

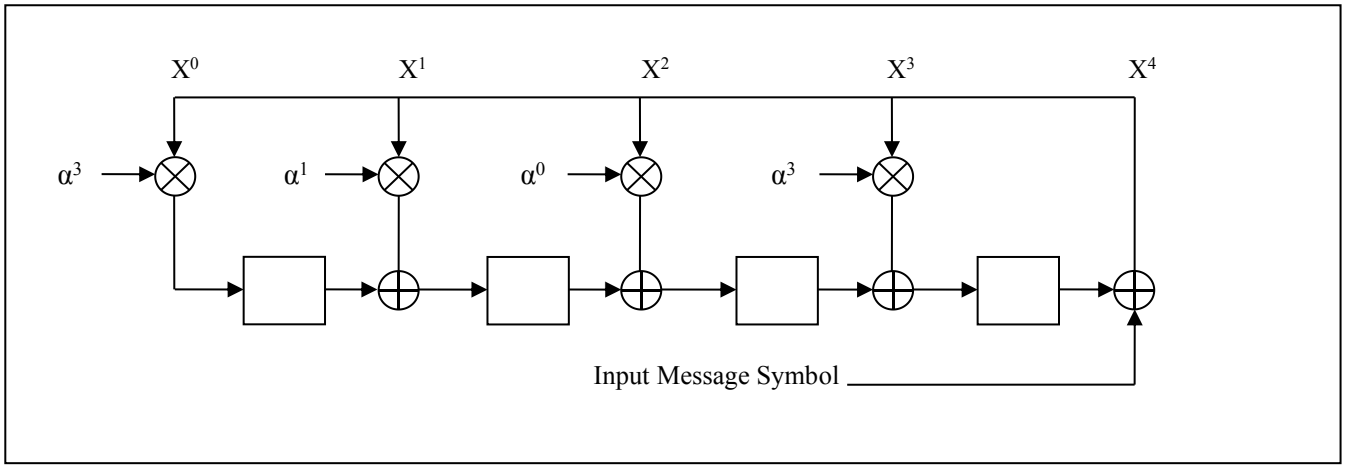

Fig. 3 LSFR for parity symbol generation

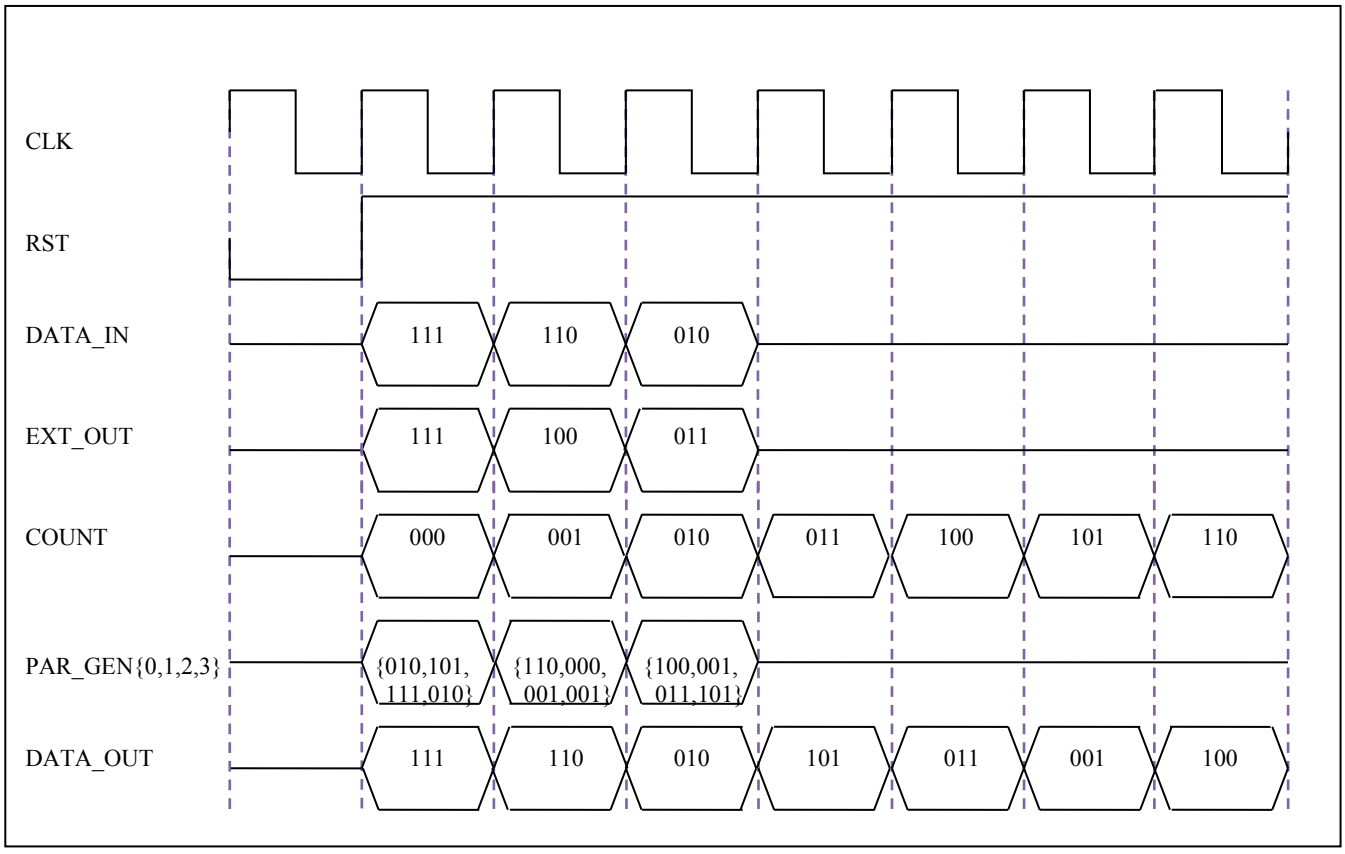

Fig. 4 Timing diagram

Depending on the data loaded into the extension field generator (*EXT_OUT*), the "Parity Generator" generates the four parity symbols based on the LSFR shown in Figure 3. '+' indicates modulo 2 addition and 'x' indicates Galois field multiplication. This parity symbol generation happens during the first three clock cycles. At the end of the third clock cycle the four parity symbols are present in the four parity generator registers.

An easier way of performing Galois field multiplication is to load one symbol to be multiplied into extension field generator and shift the generator as many times as the second symbol. For example, if the two symbols are α^6 and α^2, then we load α^6 into the generator and shift it twice cyclically to right, resulting in α^8 which is α^1 (modulo 7).

Parity generator is implemented as a quasi-look up table. That is, the data into the generator (*EXT_OUT*) is pre-multiplied with the multiplicands at the respective stages (Fig. 3) and then XORed. The resulting output is fed into output generator which output's parity symbols after three clock cycles.

B. Timing Diagram

The timing diagram of the encoder is given in figure 4. It shows the various events happening in the encoder with respect to clock. The sub modules are sensitive to the positive edge of the clock (*CLK*). The system uses a negative reset (*RST*). That is, when the reset is 1 the normal operation of the encoder begins. The data to be loaded into the extension field generator (*EXT_OUT*) is got by XORing the input data (*DATA_IN*) with the contents of the fourth stage LSFR register (Fig. 3). The counter counts from 0 to 6 (*COUNT*), then resets to 0 again. The input data is available for only three clock cycles because k = 3. During each of the three clock cycles, the parity symbols are generated in the LSFR (Fig. 3). The final parity values are got after three clock cycles. The output data (*DATA_OUT*) is same as the input during the first three clock cycles after reset. During the next four clock cycles, it is the parity symbols.

VI. Results and Observations

The frequency of operation of the encoder module and the slices utilized are tabulated in Table I for various FPGA families from Xilinx. These values are compared with other well known Reed-Solomon encoder designs from various IP core vendors, namely HITECH GLOBAL, VXL Technologies and AVNET Engineering Services.

TABLE I
Reed-Solomon (7, 3) Encoder Performance Parameters

Family	Spartan 3E	Virtex2	Virtex4	Virtex4	Virtex5
Device	XC3S 1200E	XC2V 40	XC4V LX25	XC4V SX35	XC5V LX30
Package	FT256	CS144	SF363	FF668	FF324
Speed Grade	-5	-6	-11	-10	-1
Frequency (MHz)					
Present Design	270.544	394.555	778.210	668.405	624.220
HITECH[6]	110	-	180	-	200
VXL[7]	-	-	-	118	-
AVNET [8]	-	-	-	-	-
Slices					
Current Design	22	22	33	33	38
HITECH[6]	1106	-	989	-	306
VXL [7]	-	-	-	2567	-
AVNET [8]	-	89	-	-	-

From the table we observe that the Encoder design presented in the current paper is superior both in terms of speed of operation and area utilization. This performance improvement is largely attributed to the simplified Galois field multiplier design.

VII. Conclusion

The Reed-Solomon (7, 3) Encoder is designed in Verilog HDL and implemented using Xilinx FPGA's. The encoder architecture is designed by taking speed and area as the performance parameters. The operation of the encoder is thoroughly explained by taking its architecture and timing diagram. A simpler method of performing Galois field multiplication is explained which is largely responsible for the improvement in performance. Since the design of the most critical module is simplified, the entire design gives higher performance. Finally, the performance parameters are compared with other well known designs.

References

[1] Stephen B. Wicker and Vijay K. Bhargava, *Reed-Solomon codes and their applications*, IEEE Press, 1994.

[2] Bernard Sklar, *Digital Communications – Fundamentals and Applications*, 2nd ed, Pearson Education.

[3] John G. Proakis, *Digital Communications*, 4th ed, McGraw – Hill International edition.

[4] Simon Haykin, *Digital Communications*, 8th ed, John Wiley & Sons Inc.

[5] I. S. Reed and G. Solomon, "Polynomial codes over certain finite fields", *SIAM Journal of Applied Mathematics*, vol. 8, pp. 300–304, 1960.

[6] (2009) HITECH GLOBAL page. [Online]. Available: http://www.hitechglobal.com/ipcores/rse.htm

[7] (2009) VXL Technologies page. [Online]. Available: http://www.ipgeniuscores.com/docs/vtl-tsi_rs_pb_v01.pdf

[8] (2009) AVNET page. [Online]. Available: http://www.em.avnet.com/ctf_shared/mgw/df2df2usa/Avnet-MGW-IP-Xilinxcorelist072007.pdf