ASIC Implementation Of Reed Solomon Codec For Burst Error Detection And Correction

Sangeeta Singh
Associate Professor
Dept. of ECE
Vardhaman College of Engineering
Hyderabad (A.P)

S. Sujana
Associate Professor
Dept. of ECE
Vardhaman College of Engineering
Hyderabad (A.P)

Abstract

Accuracy of information in any communication system is very critical. Use of Forward Error Correction (FEC) to lower the probability of error and increase transmission distance has become widespread. Reed-Solomon code is one of the block FEC, capable of correcting multiple errors, focusing specifically on burst errors, making it popular for mass storage devices, wireless and mobile communication units, digital television (DTV), satellite communications, digital video broadcasting (DVB) and broadband modems. When RS(n, k) codes are used for high reliable systems, the occurrence of faults in the encoder and decoder subsystems should be considered. Reed Solomon codec consists of both encoder and decoder on a single chip. In this paper new architecture is proposed for RS codec which consists of an encoder, decoder and a noise block that generates random noise. The RS encoder architecture is designed using LFSRs which exploits some properties of the arithmetic operations in G(2^m). The Reed-Solomon decoder processes each block and attempts to correct up to t = (n-k)/2 symbols and recovers the original data. In the RS decoder, the implicit redundancy of the received codeword, under certain assumptions explained in this paper, allows implementation of concurrent error detection schemes useful for a wide range of applications. In this paper new decoding method for Reed Solomon codec is proposed which uses Berlekamp's algorithm instead of Euclidean method. The codec is designed using verilog hardware description language and simulated using Xilinx tools. To improve the performance of the RS codec, the same design is synthesized and implemented with 180nm TSMC library using cadence tools.

1. Introduction

Wireless technology is fast becoming a trend in present communication systems. The demand for greater bandwidth allocation is being addressed by fixed wireless broadband access. However, the use of free space, as a transmission medium, introduces many sources of error in the data being transmitted across the channel. As the accuracy of information is very critical, the use of Forward Error Correction (FEC) methods has gained tremendous importance. FEC improves the reliability of data reception for a system. The basic principle behind any error correcting codes is the application of a mathematical transform onto the message signal such that redundant message information is used to correct any errors that may have been introduced during transmission.

In the design of high reliable electronics systems both the Reed-Solomon (RS) encoder and decoder should be self checking in order to avoid faults in these blocks which compromise the reliability of the whole system. In fact, a fault in the encoder can produce a non correct codeword, while a fault in the decoder can give a wrong data word even if no errors occur in the codeword transmission. Therefore, great attention must be paid to detect and recover faults in the encoding and decoding circuitry. Nowadays, the most used error correcting codes are the RS codes, based on the properties of the finite field arithmetic. In particular, finite fields with 2^m elements are suitable for digital implementations due to the isomorphism between the addition, performed modulo 2, and the XOR operation between the bits representing the elements of the field.

The use of the XOR operation in addition and multiplication allows to use parity check-based strategies to check the presence of faults in the RS encoder, while the implicit redundancy in the code-word is used either for correct erroneous data and for detect faults inside the decoder block.

2. RS Codes

RS codes are an example of a block coding technique, where the data stream to be transmitted is broken up into blocks and redundant data is then added to each block. The size of these blocks and the amount of check data added to each block is either specified for a particular application or can be user-defined for a closed system. RS codes are specified using (n, k) notation where 'n' represents the total length of the codeword and 'k' refers to the number of original message symbols of m-bit each.

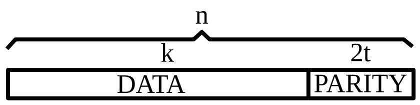

Figure 1: *RS Codeword*

The advantage of using Reed Solomon codes is that it can correct multiple errors. In general there are (n-k) parity symbols of 'm' bits each. A Reed-Solomon decoder can correct up to 't' symbols that contain errors in a codeword, where 2t = n-k. It is mainly used to correct burst errors in mass storage devices, communication systems, digital video broad-casting (DVB) and broadband modems. The amount of processing "power" required to encode and decode Reed-Solomon codes is related to the number of parity symbols per codeword. A large value of 't' means that a large number of errors can be corrected but requires more computational power than a small value of 't'.

The finite fields used in digital implementations are in the form GF(2^m), where m represents the number of bits of a symbol to be coded. More information about finite fields and RS codes are provided in [1]. An element a(x) $\in$ GF(2^m) is a polynomial with coefficients $a_i \in \{0,1\}$ and can be seen as a symbol of m bits $a=a_{m-1} \ldots\ldots a_1 a_0$. The addition of two elements a(x) and b(x) ϵ GF(2^m) is the sum modulo 2 of the coefficients a_i and b_i, i.e., is the bitwise XOR of the two symbols a and b. The multiplication of two elements a(x) and b(x) ϵ GF(2^m) requires the multiplication of the two polynomials followed by the reduction modulo i(x), where i(x) is an irreducible polynomial of degree m. Multiplication can be implemented as an AND-XOR network, as explained in [5].

3. RS Encoder and Decoder

The RS(n, k) code is defined by representing the data word symbols as elements of the field GF(2^m) and the overall data word is treated as a polynomial d(x) of degree k-1 with coefficient in GF(2^m). The RS codeword is then generated by using the generator polynomial g(x). All valid code words are exactly divisible by g(x). The general form of g(x) is

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

$$g(x) = g_0 + g_1 x + g_2 x^2 + \ldots\ldots g_{2t-1}x^{2t-1} + x^{2t}$$

where 2t = n-k, α = primitive element.

The codeword's of a separable RS(n,k) code correspond to the polynomial C(X) with degree n-1 that can be generated by

$$P(x) = d(x).x^{n-k} \operatorname{mod} g(x)$$

$$c(x) = p(x) + x^{n-k} m(x)$$

Where p(x) is a polynomial representing the parity symbols. In general encoder takes 'k' data symbols and adds '2t' parity symbols obtaining 'n' symbol codeword. The '2t' parity symbols allows correction of up to 't' symbols containing errors in a code word. From a device utilization standpoint, the size of the encoder is most heavily affected by the number of check symbols required for the target RS code. The total message length, as well as the field polynomial and first root value, do not have any appreciable effect on the device utilization or performance for a given target RS code.

In digital hardware, the encoder block is implemented using an LFSR with internal feedback connections corresponding to g(x).The computation of the remainder is implemented on digital hardware using a linear feedback shift register configuration as shown in Figure 2. Note that this setup resembles the iterative method of polynomial division. The final contents of the shift registers will contain the remainder.

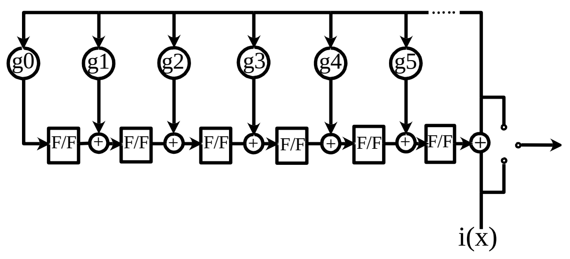

Figure2: *Encoder Architecture*

When a received block is given as input to the decoder for processing, the decoder first verifies whether the received block appears in the dictionary of valid code words. If it does not, then errors must have occurred during transmission. This part of the decoder processing is called error detection. The parameters required to reconstruct the original encoded block are available to the decoder. The decoder attempts to perform reconstruction if errors are detected. This is called *error correction*. Conventionally, decoding is performed by the Petersen-Gorenstein-Zierler (PGZ) algorithm, which consists of following parts:

i. Syndromes calculation.
ii. Derivation of the error-location polynomial.
iii. Error Locations
iv. Error Magnitudes
v. Error Correct

The RS decoder consists of five major blocks as shown below:

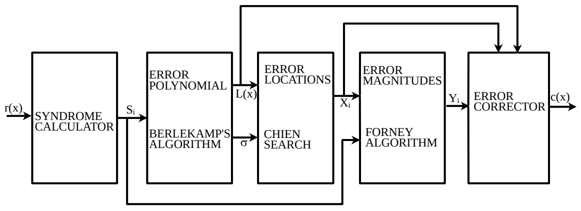

Figure 3: *Decoder Architecture*

In this implementation the error-location polynomial is found using the Berlekamp-Massey algorithm, and the error values are obtained by the Forney algorithm.

Let the received codeword R(X) :

R(X) = C(X) + E(X)

Where C(X) = original (transmitted) codeword, E(X) = error polynomial

A codeword's syndrome *s(x)* is the remainder of the division of the received word *r(x)* by the generator polynomial, as implied by the following equation:

$$\frac{r(x)}{g(x)} = q(x) + \frac{s(x)}{g(x)}$$

$$\frac{c(x)+e(x)}{g(x)} = q(x) + \frac{s(x)}{g(x)}$$

Since all code words are divisible by the generator polynomial, only the error component will yield a remainder.

$$\frac{e(x)}{g(x)} = q_e(x) + \frac{s(x)}{g(x)}$$

The above equation shows that the syndrome is independent of the message information and depends only on the error component. For no errors, the syndrome polynomial *S(x)* will be zero. In most systems, partial syndromes are computed instead of the syndrome, for reasons of simpler hardware implementation. In the computation of a partial syndrome, the divisor is no longer the entire generator polynomial, but only one of its factors, as shown in below equation:

$$\frac{r(x)}{(x-a_k)} = q(x) + \frac{s_k}{(x-a_k)}$$

There will be *n-k* partial syndromes for every received word, since the generator polynomial has *n-k* factors. The hardware implementation is shown Figure 4.

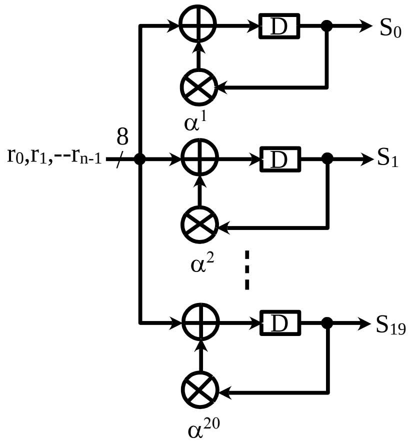

Figure 4: *Syndrome Calculation*

The method used for deriving error locator polynomial in this implementation is the Berlekamp-Massey Algorithm[6]. The Berlekamp-Massey algorithm is a shift-register synthesis algorithm which takes the n-k partial syndromes as input and outputs the error locator polynomial σ(X). The Berlekamp Massey algorithm is an algorithm for finding the shortest linear feedback shift register (LFSR) for a given output sequence.

The roots of error locator polynomial provides the error locations and is obtained by performing the Chien Search, which evaluates the Error Locator Polynomial at all elements of the GF(2^m) field. The algorithm checks if σ(α^P)equals zero, p = 0, 1, 2 …., n, then α^P is a root of the polynomial, and α^P is an error location, X_P. This is implemented using LFSRs similar to those used in computing the partial syndromes. The equation that determines the error evaluator or error magnitude polynomial Ω(X) is given by

$$S(X)\sigma(X) = \Omega(X) \bmod X^{n-k}$$

An efficient way of computing Ω(X) is to perform parallel computation of σ(X). The Forney Algorithm is used to compute for the error magnitudes, Y_i, corresponding to the respective error locations using the following equation:

$$Y_i = \frac{\Omega\left(X_i^{-1}\right)}{\sigma'\left(X_i^{-1}\right)}$$

where X_i^{-1} indicates the root as computed from the Chien Search, and $\sigma^1(X)$the derivative of the error locator polynomial.

The error corrector block takes the received code and performs XOR-operation with the corresponding error magnitudes computed at the respective error locations to attain the original message stream.

$$C(X_i) = R(X_i) \oplus Y_i$$

4. Results

In this paper the top module of the design integrates data-rom block for input, an encoder, noise block and decoder. The design receives message symbols using data-rom that generates sequence of symbols for input. These are processed by encoder operating on Galois Field to form a complete code word by adding parity symbols. A noise block generates random noise that gets added to the encoded message and converts it into an incorrect information. The function of decoder is to detect all possible errors and correct them.

The design is implemented in verilog hardware description language and simulated using Xilinx on Spartan-3E. RS Codec is further synthesized using 180 nm TSMC technology. Table-1 shows the results obtained for the order RS(208, 200) using Xilinx.

Timing Delay Report: This gives the delays that will be present in the realization of the design once it is implemented on a FPGA kit. It is the sum of the logic delays and the wiring delays. The wiring delays must be kept as low as possible and at times are comparable to the logic delays. The total delay is thus the sum of all the logic delays involved and the wiring delays. This is depicted in Table-2 which shows the delays involved in the FPGA implementation of our design.

Table 1: *Device Utilization Summary*

Device Utilization Summary Selected Device:3x500efg320-4		
Number of Slices:	1446 out of 4656	31%
Flip Flops:	671 out of 9312	7%
Number of 4 input LUTs:	2635 out of 9312	28%
Number of bonded IOBs:	22 out of 232	9%
Number of GCLKs:	1 out of 24	4%

RTL compiler generates schematic for each sub modules including the top module after completing synthesis. Figure 5 to 7 show RTL schematic of top module, encoder and input message module respectively.

Table 2: *Timing Summary.*

Timing Summary Speed Grade: -4	
Minimum period:	24.379ns (Maximum Frequency: 41.019MHz)
Minimum input arrival time before clock:	13.201ns
Maximum output required time after clock:	28.420ns

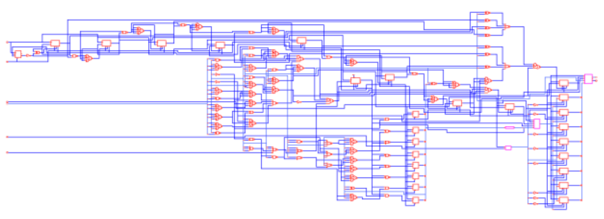
Figure 5: *Top Module of RS(208,200) Codec*

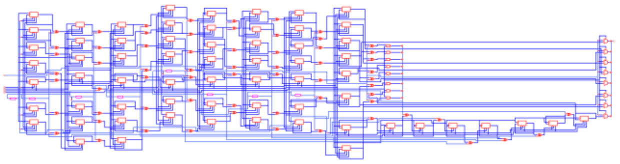
Figure 6: *Encoder Module*

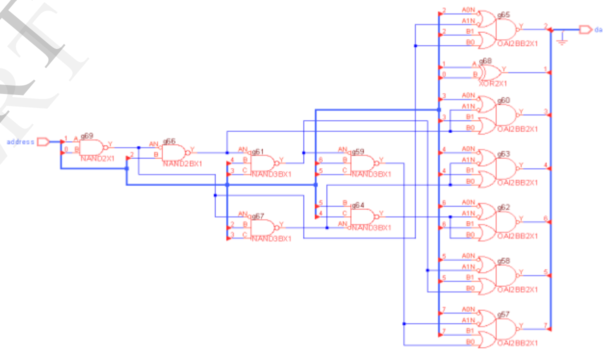
Figure 7: *Input Message Module*

The same design is implemented using SOC encounter tool. Reports on gate count, power, timing etc are generated at the output. The leakage power obtained is 750.127 nW .

Type	Instances	Area	Area %
Sequential Inverter Buffer Logic	649 236 7 4326	46842.365 1609.978 93.139 84207.816	35.3 1.2 0.1 63.4
Total	5218	132753.298	100

Table 3: *Gates Report*

5. Conclusion

Reliability of information is critical in any communication systems. Use of error-control codes reduces interference effects, and FECs in general, eliminate the need for retransmission of data streams. RS (208,200) Codec is capable of correcting 4 errors at a time and is mainly used to correct burst errors in storage devices.

The design has been verified using Xilinx tools on Spartan-3E FPGA where the operating frequency is 41.019 MHz and the same is also synthesized using 180nm TSMC library RTL compiler and SOC Encounter to improve the performance by increasing the clock frequency to 100 MHz.

The RS Codec designed here considers only pseudo random noise. The same design can be extended to model different types of noises like Poisson's distribution, Gaussian distribution. The length of the codeword can be increased in order to correct more number of errors.

6. References

[1] R. E. Blahut, Theory and Practice of Error Control Codes. Reading, MA: Addison-Wesley Publishing Company, 1983.

[2] S. B. Wicker, Error control Systems for Digital Communication and Storage, Prentice Hall, 1995 G.C

[3] G.B. Agnew, T. Beth, R.C. Mullin, and S.A. Vanstone,"Arithmetic Operations in GF(2^m)," J. Cryptology, vol.6, pp. 3-13,1993.

[4] Candarilli, S.Pontarelli, "Concurrent Error Detection in Reed-Solomon Encoders and Decoders"- IEEE trans. VLSI Systems. , Volume 15, July 2007.

[5] A. R. Masoleh and M. A. Hasan, "Low complexity bit parallel architectures for polynomial basis multiplication over GF(2^m), computers," IEEE Trans. Computer., vol. 53, no. 8, pp. 945–959, Aug. 2004.

[6] G. L. Feng and K. K. Tzeng, "A generalization of the Berlekamp-Massey algorithm for multisequence shift register synthesis with application to decoding cyclic codes," IEEE Trans. Inform.Theory, volume. 37,pp. 1274–1287, 1991.

[7] W.J. Ebel, W. H. Tranter, The Performance of Reed-Solomon Codes on a Bursty-Noise Channel, IEEE Transactions on Communications, Vol. 43, No. 2/3/4, February/March/April 1995.

[8] S. P. Kang, C. G. Kim, S. W. Rhee, and Y. Jee, "ASIC Implementation of Reed-Solomon Error Correction Circuits for Low Area Overhead on Memory System," International Conference on Electronics, Information, and Communication (ICEIC 2008), pp. 339-342, June. 2008.

[9] M. Gossel, S. Fenn, and D. Taylor, "On-line error detection for finite field multipliers," in Proc. IEEE Int. Symp. Defect Fault Tolerance VLSI Syst., 1997, pp. 307–311.

[10] T.A. Gulliver, M. Serra, and V.K. Bhargava, "The Generation of Primitive Polynomials in GF(q) with Independent Roots and Their Application for Power Residue Codes, VLSI Testing and Finite Field Multipliers Using Normal Bases," Int'l J. Electronics, vol. 71, no. 4, pp. 559-576, 1991.

[11] S. B. Sarmadi and M. A. Hasan, "Concurrent error detection of polynomial basis multiplication over extension fields using a multiple-bit parity scheme," in Proc. IEEE Int. Symp. Defect Fault Tolerance VLSI Syst., 2005, pp. 102–110.

[12] E.D. Mastrovito, "VLSI Architectures for Computation in Galois Fields," PhD thesis, Linkoping Univ., Linkoping, Sweden, 1991.

[13] M.A. Hasan, M.Z. Wang, and V.K. Bhargava, "Modular Construction of Low Complexity Parallel Multipliers for a Class of Finite Fields GF(2^m)," IEEE Trans. Computers, vol. 41, no. 8, pp. 962-971, Aug. 1992.