Study of full parallel RS(31,27) encoder for a 3.2 Gbps serial transmitter in 0.18 µm CMOS technology

G. Zhang, Q. Sun, T. Liu, D. Gong, B. You, L. Xiao, X. Sun, J. Ye, D. Yang, Y. Feng and J. Wang

State Key Laboratory of Particle Detection and Electronics, University of Science and Technology of China, Hefei, Anhui 230026, China

Department of Physics, Southern Methodist University, Dallas, TX 75275, U.S.A.

Department of Physics, Central China Normal University, Wuhan, Hubei 430079, China

E-mail: wangjian@ustc.edu.cn

ABSTRACT: This work presents the design of an RS(31,27) Reed Solomon encoder for a 3.2 Gbps serial transmitter in 0.18 µm CMOS technology. The RS encoder is used to protect the data transmission from being corrupted by particle radiation. The proposed encoder is designed with a novel full parallel structure optimized for high speed and high stability. One data frame contains 2 interleaved RS(31,27) codes and thus it can correct at most 20 bits of consecutive errors. A corresponding decoder is implemented on Xilinx Kintex-7 FPGA. The decoder algorithm is Berlekamp-Massey algorithm.

KEYWORDS: Digital electronic circuits; Pixelated detectors and associated VLSI electronics; VLSI circuits

Corresponding author.
1 Introduction

1.1 FEC code introduction

Reed Solomon code is one of the most powerful forward error correction (FEC) code [1]. It can improve the transmission accuracy with low overhead. It is even suitable for burst error correction, which is common in optical fiber transmission in radiation environment. Reed Solomon code is defined in finite field $GF(2^m)$. Each symbol has m bits. Reed Solomon code can be represented as RS($N,K$), where $N$ means the length of code, and $K$ means the length of data. $N$ is usually be set to $2^m - 1$. RS code is cyclic code. It can be constructed with a generator polynomial defined in $GF(2^m)$. The constructed code needs to be divisible by the generator polynomial. As a cyclic code, the encoder of RS code can be implemented as a linear feedback shift register (LFSR) hardware structure. The decoding algorithms of RS code are divided into two classes: soft-decision decoding algorithm and hard-decision decoding algorithm. In this project hard-decision algorithm is used, since the receiver cannot extract soft information from the communication channel. Berlekamp-Massey algorithm [2] is an algebra based hard-decision decoding algorithm. It can correct at most $(N - K)/2$ symbols exactly.

A low-density parity-check (LDPC) code is another kind of FEC code. Being compared with Reed Solomon code, LDPC code is more complex on both encoding and decoding sides. LDPC can be divided into two kinds: binary LDPC and non-binary LDPC. Like Reed Solomon code, non-binary LDPC code is also defined in finite field. The difference is that both binary and non-binary LDPC codes are constructed with generator matrix. A random chosen generator matrix requires a very big memory to store it. Another kind of generator matrix is some shifted submatrix. The LDPC code generated with such a generator matrix is also called quasi-cyclic LDPC (QC-LDPC). Although QC-LDPC doesn’t require a big memory, it has fairly strict limitation on the code length.
and data ratio, so it cannot be used to construct a short, high-data-ratio code. LDPC code has a lot of high performance soft-decision decoding algorithms and also a few of hard-decision decoding algorithms. LDPC performs well in analog communication channel. IHRB-MLGD [3] is a hard-decision decoding algorithm for non-binary LDPC code. It is a kind of iteration algorithm based on reliability. The memory requirement of the hardware implementation is O(nq). The correction performance of IHRB-MLGD with 50 iterations is almost the same with Reed Solomon, while the complexity is much higher than the latter [3].

1.2 RS(31,27) code

| Code               | Performance | Complexity | Decision |
|--------------------|-------------|------------|----------|
| Reed Solomon       | Medium      | Medium     | Hard     |
| LDPC+bit-flip      | Low         | Medium     | Hard     |
| NB-LDPC+IHRB       | Low         | High       | Hard     |
| LDPC+bp            | High        | High       | Soft     |

Table 1. Comparison of FEC codes.

Considering the encoder complexity and the correction performance (table 1), we choose RS(31,27) as the FEC code of the transmitter. The RS(31,27) code has 27 information symbols and 4 redundancy symbols. Each symbol contains 5 bits. There are totally 155 bits in one code, including 135 information bits and 20 redundancy bits. The generator polynomial of this RS code is:

\[ g(X) = (X - \alpha^{m_0})(X - \alpha^{m_0+1}) \cdots (X - \alpha^{m_0+2t-1}) \]  

We choose \( m_0 = 27 \) in this design. This generator polynomial is specifically chosen for optimizing the decoder hardware structure. The detail of this optimization is introduced in section 3.

One data frame is constructed with 10-bit header and two interleaved RS(31,27) codes, so this codec can correct at most 20 bits of consecutive errors. Each frame contains 320 bits. One must be generated in one 10 MHz clock cycle to reach the speed of 3.2 GHz.

The transmitter is designed for data transmission of a monolithic active pixel sensor (MAPS) [4] in the ATLAS experiment. Triple modular redundancy (TMR) has been implemented to protect the logic from single event upset (SEU). The sensor data is sent from the transmitter ASIC through optical fiber and goes to the decoder running on a Xilinx Kintex-7 FPGA board. When a charged particle hits the optical-to-electrical transducer, it will cause consecutive burst error. The receiver should detect and try to correct the error code. If there is no error or it corrects the data successfully, it outputs the correct data. If it fails to correct, it outputs a failure flag with the original received data. The failed data will be discarded finally.

2 Encoder design

We firstly tried the regular serial structure and a novel parallel structure to develop the RS encoder. The serial encoder structure in figure 1 is a LFSR based encoder structure, which is common in many existing designs [5]. The input data port width is 5 bits. This encoder structure needs 31 clock
cycles to encode one set of data. The driving clock of the encoder is set to 320 MHz to meet the speed requirement. All of the additions and multiplications are performed in finite field GF(32) \[6\]. The encoder has to complete a multiplex, a GF(32) multiplication and a GF(32) addition in one cycle (3.125 ns). The circuit cannot meet the timing constraints in our 0.18 \(\mu\) CMOS technology. A way to solve the timing issue is to add another pair of encoders to the transmitter to slow down the clock speed requirement, and a data buffer and an output multiplex are also needed for data routing. This will complicate the design.

Instead of adding another pair of encoders, we developed a novel parallel encoder structure shown in figure 2. The relationship between information bits and validation bits can be calculated using symbolic computation. A python script uses manual designed symbolic computation to simulate the verilog encoder circuit, runs it for 31 cycles and outputs the generated verilog mapping circuit. The python script takes advantage of the Low Complexity Bit Parallel (LCBP) \[7\] algorithm to calculate multiplication in finite field. The multiplication between constant number and register can be done with pure combinatorial logic and only needs XOR gate, so the generated circuit is completely a combinatorial logic circuit composed with only XOR operations. The encoder needs only one cycle to encode one code with the clock speed of 10 MHz. Each validation bit is a XOR chain of information bits. The longest XOR chain contains XOR operations for 70 information registers. The XOR chains are synthesized as a tree of XOR3 gates, so the time complexity is \(O(\log_3 N)\), and the deepest tree has a depth of 4. The parallel encoder can be speed up by just increasing clock frequency with no extra resource increment. Our scrambler is designed as pure combinatorial logic, so the longest timing path is from scrambler input to encoder output. The delay of scrambler is 2.37 ns, and the delay of the longest encoder path is 4.67 ns. Considering the clock network delay, this design can work under 100 MHz clock.

The parallel structure costs about 5 times resources than the serial structure. The serial encoder needs to be duplicated to reduce clock frequency, which increases complexity and delay. In consideration of the duplication and the multiplex of the serial encoder, the resource quantity of the parallel structure is not much more than the serial encoder. The serial encoder also needs two data width converters, from 135 bits to 5 bits and from 5 bits to 155 bits. The benefit of parallel structure is that the fastest clock speed is only 10 MHz, so it is fairly simple to get timing converged and is suitable for a sub-micron CMOS technology. The parallel structure is a good approach for those who use a short Reed Solomon code and need a very fast speed.

The transmitter has been tested together with the receiver board connected with optical fiber.

![Figure 1. LFSR based encoder structure.](image)
Figure 2. Parallel encoder structure.

Table 2. Resource usage of the two encoder structures.

|                      | Parallel Encoder | Serial Encoder |
|----------------------|------------------|----------------|
| Total Cells          | 1005             | 203            |
| Combinational Cells  | 850              | 171            |
| Sequential Cells     | 155              | 32             |

3 Receiver design

A receiver with an RS(31,27) decoder is developed on a Xilinx Kintex-7 FPGA board using the Berlekamp-Massey [8] algorithm (figure 3). There are totally 4 decoder modules to process the input data. Each two modules form a group. A multiplexer switches the two groups. The clock frequency of a decoder module can be any frequency higher than 200 MHz. We choose 240 MHz in our design. A total of 40 clock cycles are needed to decode a code.

Figure 3. Receiver structure.

The decoder consists of four parts: syndrome calculator, Berlekamp-Massey equation solver, Chien search [9] and Forney Formula [10] calculator. A syndrome calculator is used to verify if the received code is divisible by the generator polynomial. According to the definition of RS code:

\[ R(X) = \sum_{i} c_i X^i = N(X)g(X) \]  \hspace{1cm} (3.1)
The syndrome calculator substitutes each root of the generator polynomial into \( R(X) \) and gets the same amount of syndrome values. The code is incorrect if not all syndrome values are zero. It will output a flag indicating if correction is needed. The calculated syndrome values are also used in the key equation solver in the next stage. Since the generator polynomial of RS(31,27) has 4 roots, the syndrome calculator has 4 child modules to verify 4 roots simultaneously. When it receives a code symbol, each child module calculates one addition and one multiplication under finite field. The finite field adder is just XOR gate; the finite field multiplier for constant and register is a full parallel structure constructed with only XOR gates [7]. The longest combinatorial path of the multiplier has 2 XOR gates. One code has 31 symbols, so the syndrome calculator needs 31 clock cycles to calculate syndrome values.

![Figure 4. Equation solver module in decoder.](image)

An equation solver is used to compute the coefficients of the Error Locator Polynomial and the Error Evaluator Polynomial. These two polynomials are used to solve the error location and the error correction value. The Berlekamp-Massey algorithm is an iterative algorithm. 4 iterations are needed to solve all coefficients of the two polynomials. The hardware implementation is the RiBM algorithm [10]. Figure 4 shows the hardware structure of the RiBM algorithm. It consists of \( 3t + 1 \) PE1 modules and a controller module. The first 4 PE1 modules are initialized with the 4 syndrome values calculated in the first stage. After iteration of 4 clock cycles, the output of the first 2 PE1 modules are the coefficients of the Error Evaluator Polynomial, and the followed 3 PE1 modules output the coefficients of the Error Locator Polynomial. The Equation solver also calculates the degree of the Error Locator Polynomial, which is used to check if the error can be corrected.

A Chien search and Forney Formula calculator can solve the error locations and the error correction values respectively, according to the two polynomials from the equation solver. The Forney Formula is:

\[
Y_i = \frac{X_i^{-(m_0+2r-1)} \Omega(X_i^{-1})}{\Lambda'(X_i^{-1})} = \frac{z^{m_0+2r} \Omega(z)}{z \Lambda'(z) \big|_{z=X_i^{-1}}} \tag{3.2}
\]

where \( m_0 \) is the parameter of generator polynomial, \( \Omega \) is the Error Evaluator Polynomial and \( \Lambda \) is the Error Locator Polynomial.
The format of $Y_i$ can be simplified when $m_0$ is set to $n-2t$, which is 27 in our case. The derivation
of $x^n$ is

$$(x^n)' = nx^{n-1} = \sum_{i=0}^{n-1} x^{n-1},$$

and

$$(x^n)' = \begin{cases} 0, & \text{if } n \text{ is even.} \\ x^{n-1}, & \text{if } n \text{ is odd.} \end{cases} \quad \text{(3.3)}$$

So only odd items need to be computed. A parallel inverse module is created to calculate the
reciprocal of $z\Lambda'(z)$.

The received data is sent to two paths, one for the syndrome calculator and the other for the
FIFO. The data moves out from FIFO when Chien search is starting searching for error locations.
The error correction value is added to the data at the same time. After Chien search is started, the
decoder is ready to receive the next group of data.

4 System structure

![Figure 5](image)

Figure 5. Transmitter structure.

Figure 5 shows the structure of the transmitter. The transmitter first scrambles sensor data
to achieve DC-balance, and then encodes the data with two RS(31,27) code encoder in parallel.
The two data streams are interleaved together to form a 310-bit data stream in 10 MHz clock. A
frame builder module prepends a 10-bit header to the data and converts it to a 32-bit data stream
in 100 MHz clock. Finally, a serializer serializes the 32-bit data stream to a 1-bit double data rate
(DDR) data stream in 1.6GHz clock.

This design was taped out with TowerJazz 180 nm CMOS process. The serializer, Phase-locked
loops (PLL) and the Current Model Logic (CML) driver were designed with analog circuits, and
the other parts were designed with digital circuits. The chip is integrated with the sensor on the
same printed circuit board.

5 Test result

Test result includes simulation test and on-board test. The simulation test is to test the encoder
and decoder Verilog code using iVerilog simulator and C programs. Two C programs of decoder
and encoder has been written to verify the encoder and decoder output respectively. After that, the
encoder and decoder connected and tested together in a Verilog test bench.

Figure 6 shows the structure of the on-board test platform. The data source is an FPGA data
generator. We also add the same data generator in the FPGA receiver to check the data correctness.
Two FINISAR FTLX8574D3BCL optical transceivers are installed for both the ASIC transmitter and the FPGA receiver. An optical attenuator is connected between the two optical transceiver. The optical attenuator is used to tune the optical modulation amplitude (OMA). The FPGA receiver uploads decoded data and the test result to PC through PCIe interface.

Figure 7 is the result of the on-board test. Each data point was measured for 10 minutes, corresponding to a total of $6 \times 10^9$ frames. We counted the accuracy of both received data and decoded data. The blue line is the error counts of decoded frames in linear Y axis, and the orange line is the error counts of received frames in logarithm Y axis. The received data is error-free when the OMA is above $-16.8 \text{ dBm}$, while the decoded data is error-free when the OMA is above $-16.9 \text{ dBm}$. The distance between orange line and blue line is the number of corrected frames. The Reed Solomon codec successfully recovers 43 thousand incorrect frames at $-17.63 \text{ dBm}$, and recovers 90 incorrect frames at $-16.9 \text{ dBm}$ [11].

![Error Counts vs. OMA](image)

**Figure 7.** RS(31,27) performance measured with the error counts before and after correction.
On-board test under radiation environment has been scheduled. The goal of this test is to check the capability of the TMR protection from the SEU effect, and also the performance of the Reed Solomon code under particle radiation burst error.

6 Conclusion

In this paper we presented a full parallel structure for RS(31,27) code encoder. This structure is suitable for high speed transmission with short code length. The test result shows that the codec works properly and can reduce the error rate.

During the test we found that the built-in error detector of Reed Solomon algorithm cannot detect all the error frames. A CRC error detector is needed for further improvement [12].

Acknowledgments

This work was supported by the National Natural Science Funds of China under Grant No. 11773026, 11728509, the Fundamental Research Funds for the Central Universities (WK2360000003, WK2030040064), the Research Funds of the State Key Laboratory of Particle Detection and Electronics.

References

[1] W. Ryan and S. Lin, Channel codes: classical and modern, Cambridge University Press (2009).
[2] E. Berlekamp, Nonbinary bch decoding (abstr.), IEEE Trans. Inform. Theory 14 (1968) 242.
[3] C.-Y. Chen, Q. Huang, C.-c. Chao and S. Lin, Two low-complexity reliability-based message-passing algorithms for decoding non-binary ldpc codes, IEEE Trans. Commun. 58 (2010) 3140.
[4] I. Perić, Active pixel sensors in high-voltage CMOS technologies for ATLAS, 2012 JINST 7 C08002.
[5] M.A. Hasan and V.K. Bhargava, Architecture for a low complexity rate-adaptive reed-solomon encoder, IEEE Trans. Comput. 44 (1995) 938.
[6] L.E. Dickson, Linear groups: With an exposition of the Galois field theory, Courier Corporation (2003).
[7] A. Reyhani-Masoleh and M.A. Hasan, Low complexity bit parallel architectures for polynomial basis multiplication over gf (2m), IEEE Trans. Comput. 53 (2004) 945.
[8] I.S. Reed and M.-T. Shih, Vlsi design of inverse-free berlekamp-massey algorithm, Proc. IEEE E 138 (1991) 295.
[9] R. Chien, Cyclic decoding procedures for bose-chaudhuri-hocquenghem codes, IEEE Trans. Inform. Theory 10 (1964) 357.
[10] G. Forney, On decoding bch codes, IEEE Trans. Inform. Theory 11 (1965) 549.
[11] Q. Sun, G. Zhang, D. Gong, B. Deng, W. Zhou, B. You et al., A 3.2-Gb/s serial link transmitter in 0.18 µ cmos technology for cmos monolithic active pixel sensors application, Nucl. Instrum. Meth. (2018) in press.
[12] C.-L. Yu, E.T. Pak and H.-M. Leung, Ecc/crc error detection and correction system, U.S. Patent 5,027,357, 1991.