Channel Encoder Optimisation for Resources, Efficiency and Speed

M S Akhtar

University College London

Abstract: One has to weigh up between the error correction capability of a channel encoder and the data transfer efficiency. Clear choices are to be made in selecting the symbol sizes and the speed of the encoder. The hardware solutions coupled with the requirement determination can help reclaim vital resources on an FPGA real estate, improve data transfer efficiency and substantially enhance the speed.

1 Introduction and the Channel Encoder Basics

A typical communication channel is shown in Figure 1. The source encoder/decoder and encryption/decryption circuits are optional. There is a degree of control over the errors occurring in the hardware but the channel is exposed to all sorts of elements; it does require an error correction capability to maintain the reliability, efficiency and speed of the communication link.

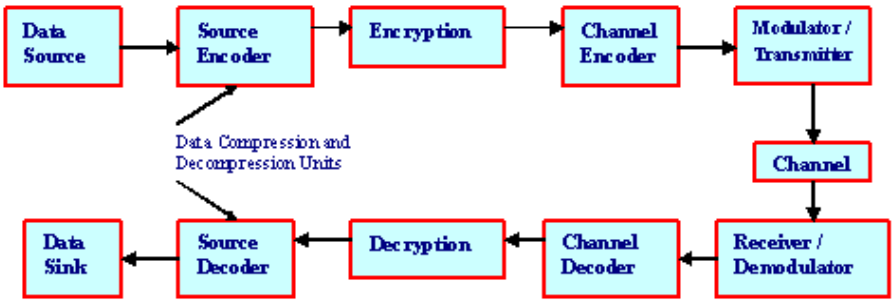

Figure 1: A typical channel encoder

The Reed-Solomon codes achieve the largest possible code minimum distance for any linear code with the same input and output block length, therefore this method for channel encoding has been selected for the discussion. This code is also known as a systematic code because the data is left unaltered and parity symbols are appended to it (Figure 2).

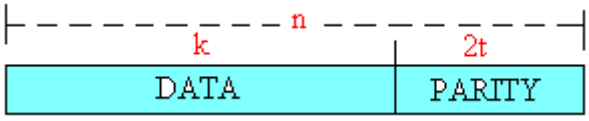

Figure 2: The coded message block where n = unaltered data symbols (k) + Parity symbols (2t)

These are non binary codes; the symbols are made up of m-bit sequences where m is any positive integer having a value greater than 1. R-S(n, k) codes on m bit symbols exist for all n and k for when $0 < k < n < 2^m + 2$, where k is the number of data symbols being encoded and n is the total number of code symbols in the encoded block[4].

$$(n, k) = (2^m - 1, 2^m - 1 - 2t) \quad \text{----------Equ1}$$

2 The Channel Encoder Performance optimization Techniques

The optimization process needs a careful analysis of requirements and considerations to weigh up the factors which constitute the optimum design. In this dialogue, design for data transfer efficiency, the speed of the data transmission and the resources saving (i.e. on FPGA real estate) are considered and an optimized design example is provided for a satellite application.

2.1 Design for Efficiency

The channel efficiency is given by $(2^m\text{-}1\text{-}2t / 2^m - 1) * 100$ % where m = symbol length and t = correctable errors in the block.

[In Figure 3; Z-axis = percentage efficiency, Y-axis = t (required correctable errors)]

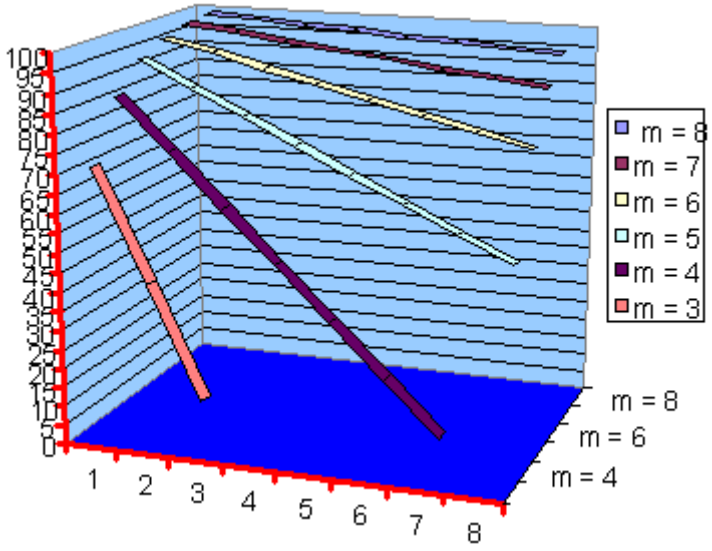

Figure 3: Channel Encoder Efficiency verses the Correctable Errors (t) for a given symbol value (m)

As illustrated in Figure 3, the efficiency is inversely proportional to the t (the correctable errors) but the slope is shallow when the symbol length is high (i.e. m = 8 gives almost flat response because of the relative size of the block comparing with the correctable errors).

2.2 Design for Speed

A Reed-Solomon codeword is generated using a generator polynomial [(g(x)].

The property of the generator polynomial [g(x)] is that it is an exact divisor for all valid code words and generally represented by Equ2.

$$g(x) = (x - \alpha^{i})(x - \alpha^{i+1})\ldots\ldots\ldots(x - \alpha^{i+2t}) \text{ ----------Equ2 [2], [1]}$$

where the order of the generator polynomial is equal to the number of error code symbols (n-k), for i = 0 the power of 'α' represents the 'on' bits in a symbol length 'm' . A parity code is generated when the message polynomial [m(x)] is divided by g(x) as shown in Equ3.

$$r(x) = m(x) \bmod g(x) \text{ (i.e. remainder)}\ldots\ldots\ldots\ldots\ldots\text{Equ3}$$

Thus a codeword can be produced by shifting the message [m(x)] by (n-k) positions and appending it to the parity code as shown below.

$$u(x) = r(x) + x^{n-k} m(x)\text{----------Equ4 [2]}$$

Equ4 is implemented by linear feedback shift registers as shown in Figure 4, for RS(7,3) and RS(15,13). In the Reed Solomon encoder the bottle neck for the operational speed is the Galois Field Multipliers in the feedback loop of the LFSR.

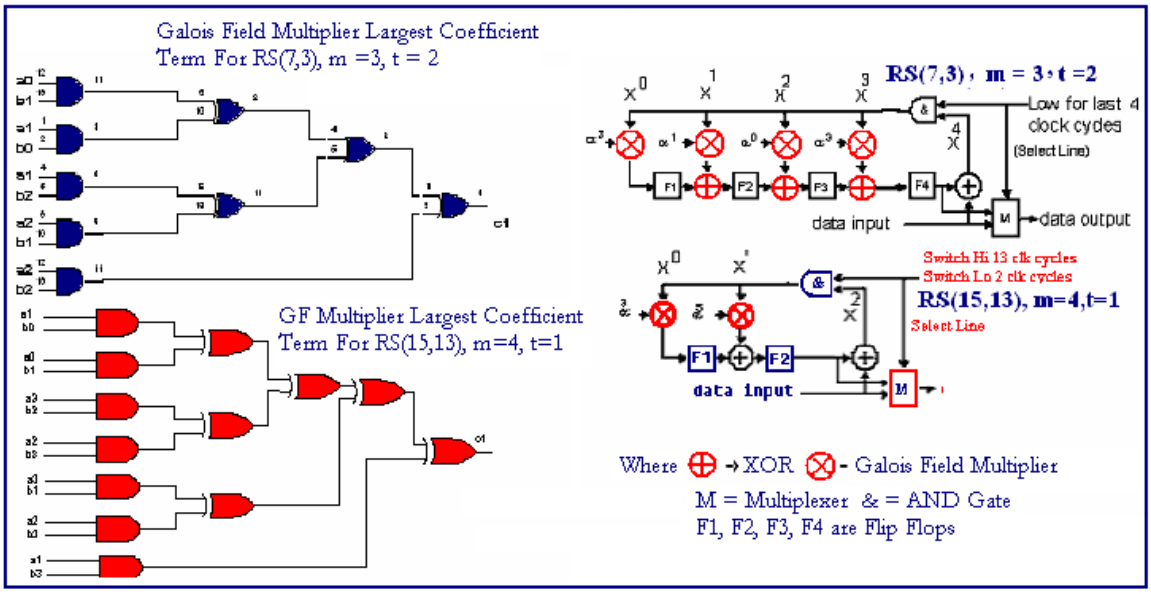

Figure 4: The Reed Solomon Encoder Implementaion Blocks and GF Multiplier Largest Coefficient

Apart from GF (Galois Field) Multipliers, the rest can be easily implemented on FPGAs. Even the GF Multiplier can be converted to combinatorial logic by the Equ7.

$$c(x) \equiv a(x) * b(x) \bmod p(x)$$----------Equ7 [3]

where a(x) and b(x) are two input polynomials and p(x) is the primitive polynomial for symbol length 'm' [2]. Two worked out examples of GF multipliers combinatorial logic for symbol lengths of m = 3 and m = 4 are shown in Equ8 and Equ9 respectively.

For m = 3: (a2 b0 + a1 b1 + a0 b2 + a2 b2) α^2 + (a1 b0 + a0 b0 + a2 b2 + a2 b1 + a1 b2) α^1 + (a2 b1 + a1 b2 + a0 b0) α^0 ----------Equ8

For m = 4: (a3 b0 + a2 b1 + a1 b2 + a0 b3 + a3 b3) α^3 + (a2 b0 + a1 b1 + a0 b2 + a3 b3 + a3 b2 + a2 b3) α^2 + (a1 b0 + a0 b1 + a3 b2 + a2 b3 + a3 b1 + a2 b2 + a1 b3) α^1 + (a0 b0 + a3 b1 + a2 b2 + a1 b3) α^0 ----------Equ9

The underlined terms (blue) are the largest coefficients in the equations above and have been drawn as the combinatorial logic gate arrays in Figure 4, which consist of largest number of gates in between the input and output of the GF Multiplier. Thus the data propagation within the GF multiplier is given by:

$$tgd * (m+1)$$ ----------Equ10

where, tgd (single gate time delay) ≈ 500 ps (for FPGAs). For example if m = 4 then the max operational frequency of the GF multiplier = 1/500 ps * 5 = (1/(2.5)ns) = 400 MHz.
Therefore the larger the symbol length, the slower the GF multiplier, hence the system speed.

2.3 Design for Resources

Referring to the LFSR blocks (Figure 4), the total estimated number of gates (XORs, ANDs) is as follows: External XORs + External ANDs + GF Multiplier gates $= (2t-1) * m + m + 2 * t * m * (2m+1) = 4 * m * t * (m+1)$ ----------Equ11

where m = symbol length and t = correctable errors.
The total number of flip flops = (2 * t * m) and the number of multiplexers (muxes) = m.

As an example; for m = 8 and t = 8, the total number of gates = 2304, the total number of flip flops = 128, total number of muxes = 8
Therefore symbol length 'm' and correctable errors't' have impact on the resources.
Note: the formulae above (Equ11) to work out the resources required are derived by numerical and observation techniques by the author.

2.4 An Optimised Design for Efficiency, Resources and Speed

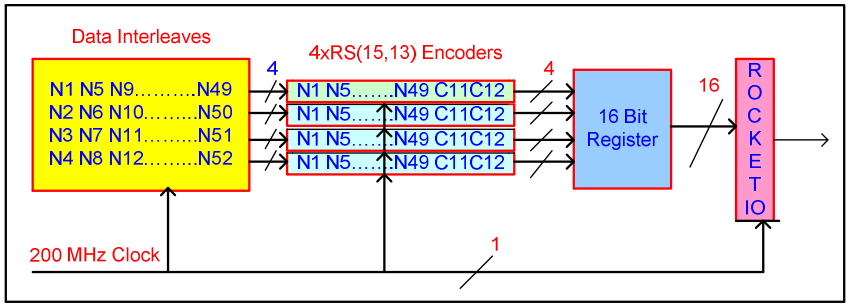

Figure 5: An Optimised Encoder Design

A RS (15, 13) hardware encoder has been implemented. This has been optimised to maximum speed possible (i.e. 400 M Symbols / second). The data bytes are split into Nibbles (N1, N2, N3….etc.) and interleaved as shown in Figure 5. A block of 13 nibbles of interleaved data is passed to each RS (15, 13) encoder, which adds two nibbles of error code to each block (i.e. C11, C12……C41, C42), thus making it a 15 Nibble block. One Nibble from each encoder is passed in parallel to a 16 bit register at each clock. The register output has a 16 bit word, ready for Rocket I/O to receive at the next clock. The Rocket I/O will transfer serial data at the rate of 200 MHz X 16 bits = 3.2 G Bits/second at 87% efficiency (i.e. for t =1).

3 Conclusion

Pre-knowledge of environmental conditions determine the ultimate optimization requirements (i.e. the EMC environment at the factory floor, SEU levels in the space or general S/N ratio). A typical SEU rate for a radiation hardened device is 10^{-8} errors/bit-day in Geo-Orbit and unhardened devices typically several orders of magnitude higher [5] [6]. The error rate could be higher when exposed to burst of SEUs. The design proposed in para2.5 is capable of correcting 4 bits (in same nibble) every 60 bits (15 nibbles) of data. Since the bit rate transfer will be 2.7 Gbits/s, thus the total time to correct a nibble or single bit in a nibble = 370ps X 60 = 22.2ns
Thus (1/[22.2ns]) ≈ 45 Mbits/second. So even SEU corrupts 45 Mbits/second, which is very unlikely, this error correcting system should still restore the error free data.

4 References

[1] C. Britton Rorabaugh ' Error Coding Cookbook' pp 9-27.
[2] Bernard Sklar, 'Digital Communications, Fundamentals And Applications' pp446-449.
[3] Gregory C Ahlquist, 'Synthesis of Small and Fast Finite Field Multiplier for Field Programmable Gate Array', School of Systems and Logistics, AirForce Institute of Technology, Wright-Patterson AFB, OH 45433.
[4] Shu Lin, Daniel J Costello, Error Control Coding/ Fundamentals and applications.
[5] R. Silberberg, "Neutron induced SEU's in the atmosphere," *IEEE Trans. on Nuclear Science*, vol. 31, no. 6, pp. 1183-1185, Dec. 1984.
[6] E.L. Petersen, "Nuclear reactions in semiconductors," *IEEE Trans. on Nuclear Science*, vol. NS-27, no. 6, pp. 1494-1499, Dec. 1980.