FPGA oriented design of parity sharing RS codecs

G.C. Cardarilli, S. Pontarelli, M.Re, A. Salsano
{pontarelli,salsano}@ing.uniroma2.it
{marco.re, g.cardarilli}@ieee.org
University of Rome "Tor Vergata", Department of Electronic Engineering
Rome, ITALY

Abstract

Reed Solomon codes are widely used to protect the information from errors in transmission and storage systems. RS codes rely on arithmetic in finite, or Galois fields. Most of the RS coders are based on the field GF(2^8), using a byte as a symbol and providing codewords up to 255 symbols. The drawback of this choice is the complexity of the arithmetic operation on the field GF(2^8), in particular with respect to fields with less number of elements, like GF(2^4). The use of the GF(2^4) field provides codewords up to 15 symbols. To supersede this limitation parity sharing RS codecs has been proposed recently, even if no hardware implementation has been provided. In this paper we analyze the hardware implementation of this kind of codes, providing a comparison with a standard RS code with the same code rate, and showing that the implementation is comparable in terms of area and there is a gain in terms of speed, in particular when the target technology is a LUT based FPGA.

I. Introduction

Error Correction Codes (ECC) are widely used to protect data transmitted on noisy channels. They are either able to detect and correct errors in the coded data stream. The choice of the ECC depends on the maximum number of errors that the code is able to detect and correct and on the performance of the encoders and decoders blocks. The performance of these blocks are mainly related to the amount of data that are able to process per unit time. Large efforts have been devoted to develop codes with increased error detection and correction capabilities, but, normally, larger is their performance greater are the efforts for developing high speed encoders and decoders architectures. Nowadays, one of the most used error correcting code class is the Reed-Solomon one. It is based on the properties of the finite field arithmetic. In particular, finite fields with a number of elements of 2^n are suitable for a digital implementation due to the isomorphism between the addition operation, that is performed modulo 2, and the XOR operation between the bits representing the field elements. Most of the RS coders are based on the field GF(2^8), using a byte as a symbol and providing codewords up to 255 symbols. The drawback of this choice is the complexity of the arithmetic operation on the field GF(2^8), in particular with respect to fields with less number of elements, like GF(2^4). On the other hand the use of more suitable Galois Field such as GF(2^4) give an upper bound to the length of the codeword of 15 symbols. To supersede this bound a multilevel parity sharing RS code has been proposed in [1] for telecoms application and in [2] for a magnetic recording systems. Even if in these papers is suggested that this kind of code is suitable for high performance hardware implementations, neither the hardware implementation nor a comparison of the obtainable performance has been driven out. In this paper we analyze the hardware implementation of this kind of codes, providing a comparison with a standard RS code with the same code rate, and shows that the implementation is comparable in terms of area and there is a gain in terms of speed, in particular when the target technology is a LUT based FPGA. The paper is organized as follows: in Section II a background of the Reed Solomon codes and parity sharing technique is presented. Section III describes the architecture proposed for the encoder, while the decoder is presented in Section IV. Some conclusions are drawn in Section V.

II. parity-sharing RS code

Reed Solomon (RS) codes are a subset of the BCH codes; moreover RS codes are linear block codes [3]. A Reed-Solomon code is specified as RS(n,k), where n represents the number of symbols of m bits (with $n \leq 2^m - 1$) of a codeword and k represents the number of symbols of the related dataword. The encoding process starts from the k data symbols (of m bits each) and adds the parity symbols to construct an n symbol codeword. Therefore, $n-k$ parity symbols are present. A RS(n,k) code is capable to correct up to $2 \cdot er + re \leq n - k$, where ϵ

Proceedings of the 2005 20th IEEE International Symposium on Defect and Fault Tolerance in VLSI Systems (DFT'05)
0-7695-2464-8/05 $20.00 © 2005 IEEE

number of erasures and re is the number of random errors. The difference between this two kind of errors is that an erased symbol is a erroneous symbol for which is know its location inside the codeword. This location is often available by processing some informations related to the transmission channel that are commonly called "channel side information". The construction of a two-level parity-sharing RS code is similar to a product code. A group of K_2 systematic (N_1, K_1) RS codewords stacked together and the last $N_1 - M$ columns are re-encoded using an (N_2, K_2) RS code. Only the first $M > K_1$ symbols of each (N_1, K_1) codeword and the $(N_1 - M) \times (N_2 - K_2)$ shared-parities are transmitted. The rate of the parity-sharing is:

$$\frac{K_1 K_2}{K_2 M + (N_1 - M)(N_2 - K_2)}$$

and a rate gain can be achieved if $N_2 < 2K_2$. Using a code RS$(2K_2, K_2)$ the rate of the parity-sharing RS code become $\frac{K_1}{N_1}$. The decoding of the two-level parity-sharing RS code is obtained by applying three steps:

1) The row codewords are first decoded to correct any erroneous symbols and the untransmitted symbols are computed like erasures.
2) The $(N_1 - M)$ column codewords are decoded to try to correct any errors and fill in the missing symbols that could not be decoded in the first row pass.
3) The rows are decoded again to fill in any remaining parities and recover the original codewords.

This algorithm allows to correctly decode the following error conditions:

1) $2er_i + re_i \leq N_1 - K_1$, where er_i and re_i are the number of erasures errors in the i row codeword, respectively.
2) $2er_c + re_c \leq N_2 - K_2$, where er_c and re_c are the number of erasures errors in a column codeword, respectively.
3) $2er' + re' \leq N_1 - K_1$, where er' is the number of rows with $2er_i + re_i > M - K_1$ and re' is the number of rows with $re_i > M - K_1$

a schema of the parity-sharing RS codeword is shown in Fig. 1.

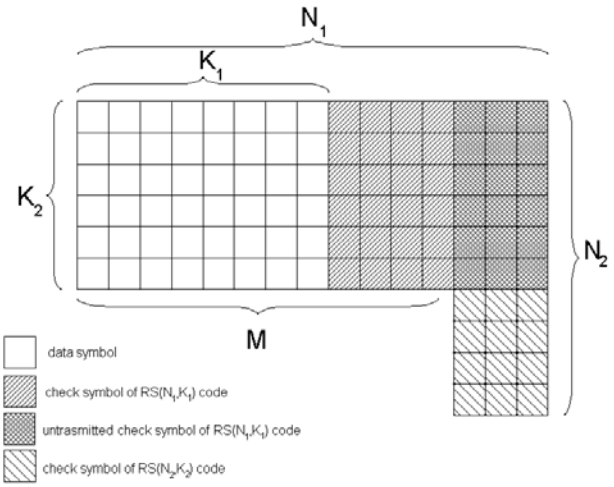

Fig. 1. Parity sharing RS Codeword

III. Encoder

In this section the design of a parity-sharing RS encoder is described. The design can be realized by using as a building blocks some RS(N_1, K_1) and RS(N_2, K_2) encoders. The implementation of a Reed Solomon encoder is usually obtained through an LFSR, which implements the polynomial division over the finite field [3]. In Fig. 2 the implementation of the RS encoder is shown.

The use of RS encoders based on the GF(2^4) allows to fully exploits the FPGA resources. The encoder is basically composed by addition and constant multiplication on the GF(2^4) field and by a set of registers. While the addition in GF(2^n) can be seen as a bit-wise xoring of the two operands, the necessary constant multipliers are mapped as generic n-inputs n-outputs logic functions. If the GF(2^4) field is used, these functions can be directly mapped by using four 4-input LUT in parallel, achieving an optimum area occupation and minimum delay.

Proceedings of the 2005 20th IEEE International Symposium on Defect and Fault Tolerance in VLSI Systems (DFT'05)
0-7695-2464-8/05 $20.00 © 2005 IEEE

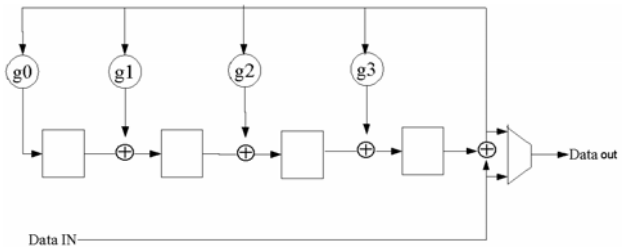

Fig. 2. RS Encoder

a constant multiplier on GF(2^8) require to use a netlist of cascaded LUTs to realize such functions, with a penalty even in terms of speed and area. In order To obtain the same data width of a standard RS(n,k) code based on a 8 bits symbol we use in parallel 2 RS(N_1, K_1) coders based on a 4 bits symbol. The codeword obtained from these coders are processed by two RS(N_2, K_2) coders. The data flow of the parity-sharing RS encoder has been designed to be similar to the RS coders one. In fact, the data flow of the standard RS encoder is presented in figure 3. The waveform shown the Data input, the corresponding output codeword and the control signals. After the data has been provided to the encoder the Ready For Data (RFD) signal is driven low in order to give out the n-k check symbols. The output information say if, on the data output bus, there is a data symbol or a check symbol, the ready output say if the data is valid, while the data valid input say to the encoder if the input data is valid.

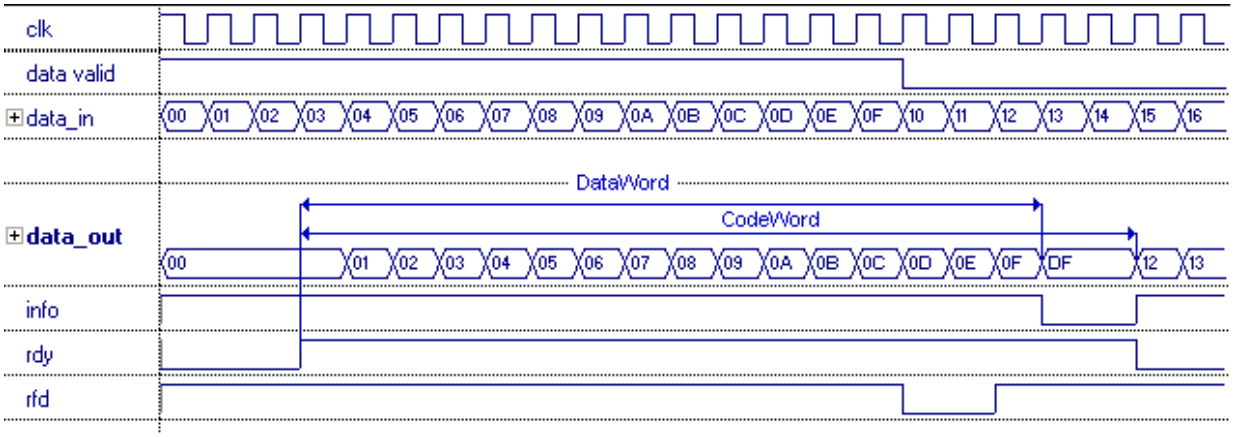

Fig. 3. RS Encoder Data flow

The proposed parity-sharing RS encoder is shown in figure 4.

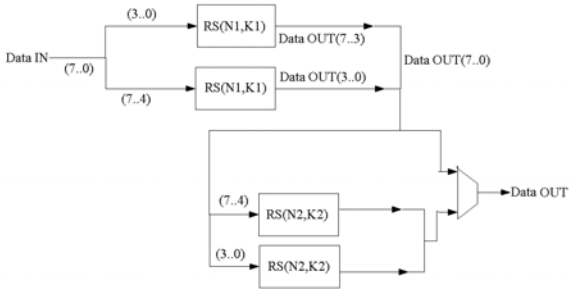

Fig. 4. parity sharing RS encoder

The control signals described above are used to control the data flow in the parity sharing RS encoder. The info output of the first encoder is used as the data valid input of the second encoder. This allows to the second encoder to process only the check symbols of the first encoder. The ready for data (RFD) output of the parity sharing encoder is the anding of the RFD of the first and of the second encoder, and the multiplexer is di

the info output of the second encoder. Table I gives the area and the speed of a standard RS coders constructed by using the IP provided by Xilinx [4] and of the building blocks of the parity sharing RS encoder. To compare the two architectures we use a RS(36,32) code versus a parity sharing code with RS(15,13) at the first level and a RS(10,6) code at the second level obtaining a similar data rate for the two compared codes. Even if a detailed comparison of the symbol error rate of the two codes is out of the scope of this paper, depending from a wide set of parameters such as the error and erasures rates, the value of n, k for the RS code and the values of N_1, K_1, N_2, K_2, M for the parity sharing code, we simply suppose that the two codes with the same data rate can give similar symbol error rates. Moreover, modifying the values of the parity sharing code only the amount of area can change, while the maximum obtainable frequency remains unchanged, depending only from the maximum delay of a single slice of the encoder. It total amount of area of the encoder can be evaluates as the sum of the building blocks. The final area occupation for the parity sharing encoder composed by two RS(N_1, K_1) and two RS(N_2, K_2) encoder is 136 slices. The area is greater than the corresponding RS encoder, but this structure can be clocked at a higher frequency 15% than the RS encoder.

Symbol Width	8	4	4
k	32	13	6
n	36	15	10
Area (Slices)	101	21	47
Latency	3	3	3
Max. Clock Frequency	285MHz	324MHz	324MHz
Xilinx Part	XC2VP-6	XC2VP2-6	XC2VP2-6

TABLE I
ENCODERS

IV. DECODER

The implementation considerations for the RS encoders can be extended to the decoder. In particular, the structure of the decoder is more complex than the encoder's one, and it requires the implementation of two additional operations: generic multiplication and element inversion on the chosen GF(2^n) field. Therefore, the consideration drawn in the previous section assume more importance in the implementation of this structure. Moreover, the development of an RS decoders with an high symbol throughput requires to analyze the time behavior of the decoders. Different timing related performance parameters must be taken into account to design an high speed RS decoder. The following parameters can be used to analyze the time behavior of the decoder:

1) Processing delay: it can be defined as the number of periods from the sampling of the first symbol of a code block, to the period in which another code block may be started. If the processing delay is greater than n, it is not possible to follow one code block immediately with another. The processing delay depends from the number of check symbols and can be evaluated by using the following formula [5].

$$Processing\ Delay = 2\left(\sum_{i=1}^{n-k-1} i\right) + 3(n-k) + 3 = (n-k)(n-k+4) + 3$$

2) Latency: it is the number of sampled symbols from a symbol being sampled on DATA_IN signal, to the corrected version of that symbol appearing on DATA_OUT. The latency is dependent on the values of n (the number of symbols in a code block) and on the number of check symbols $n-k$. It can be evaluated by using the following equation [5].

$$Latency = n + Processing\ Delay + 7$$

3) Maximum Clock Frequency: it is the maximum frequency at which the circuit can be clocked
4) Symbol Throughput: it represents the amount of data that can be processed by the decoder in a unit of time. If the Processing Delay is less than the length of the codeword, the decoder can run in continuous mode, and therefore the decoder can provide a symbol each clock cycle. Otherwise, a pause between two codewords must be inserted, deasserting the ready signal. The Symbol Throughput can be is estimated as follows:

$$Symbol\ Throughput = \frac{Max\ Freq.\ \cdot n}{Processing\ Delay}$$

where n is the number of symbols in the codeword. In the case of the RS decoder shown in Table II the symbol throughput is about 1.08 Gbits/sec.

The data flow of the standard RS decoder is shown in figure 4. The waveform shows the input codeword, the

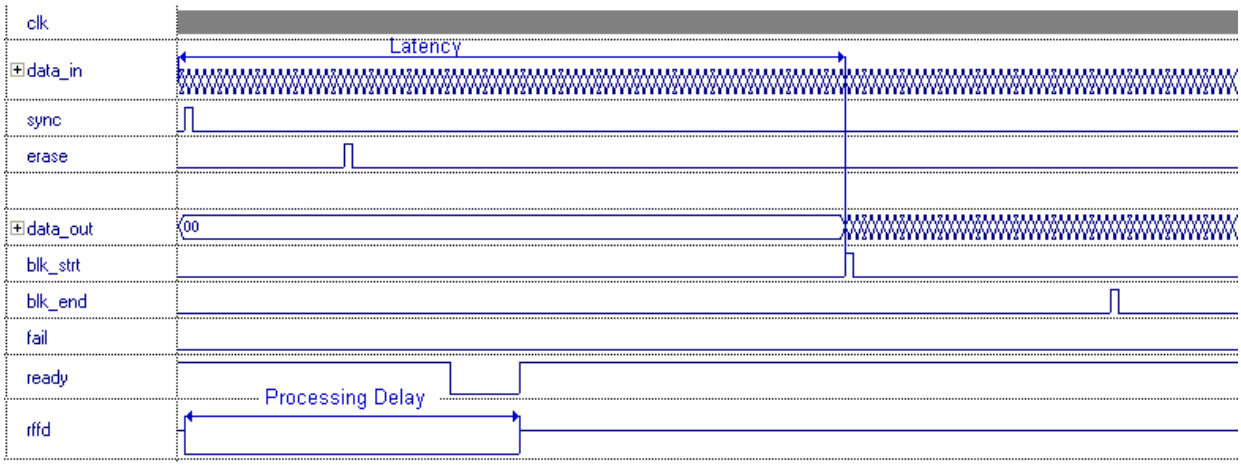

Fig. 5. RS Decoder Data flow

corresponding output codeword and the control and status signals, with the latency and processing delay. The sync input is driven high to assert that the the first symbol of the codeword is available on the data_in bus. The erasure input is driven high when the transmitted symbol is detected as an erasure. This control signal comes from a block processing the "channel side informations" available for the transmission channel. The corrected output codeword is available on the data_out bus during the interval between the assertion of the block start signal (blk_strt) and the assertion of the block end signal (blk_end). The ready status output is used to advice if the decoder can process another symbol, while the Ready For First data signal (RFFD) is used to advice if a new codeword can be processed. Finally, the fail output signal is driven high if the decoder is unable to correct the codeword due to a number of erasures and errors greater than the maximum allowed. The control and status signal of the RS decoder can be used to build a parity sharing decoder using the schema drawn in figure 5.

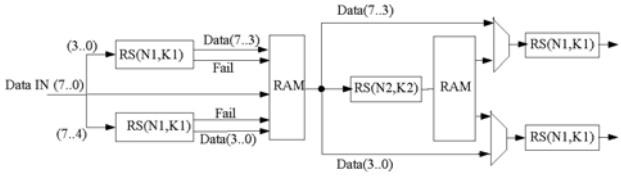

Fig. 6. parity sharing RS decoder

The first pair of RS(N_1,K_1) decoders are used to perform the first step of the algorithm described in Section II, namely the first row decoding of the codeword. The corrected codewords and the Fail flag are stored into the RAM. During this first step the erasure control signal of the first RS decoder is driven high if an erasure is detected during the transmission of the M symbols of the RS(N_1,K_1) and during $N_1 - M$ clock cycles to manage as erasures the untransmitted symbols of the RS(N_1,K_1) codeword. The second RS(N_2,K_2) decoder is used to perform the second step of the algorithm, namely the column decoding. In this step the input data are the transmitted check symbol and the decoded check symbols coming from the first pair of decoders. During the processing of the symbol coming from the first decoding the erasure control signal is driven by the fail output of the first decoder. In this way the correctly decoded check symbols are processed as good symbols, while the uncorrectable decoded check symbol are processed like erasures. Instead, during the processing of symbols coming from the transmitted codeword the erasure input is driven by the block processing the "channel side informations". After the

step of the algorithm, the corrected check symbols are re-written in the RAM and the third phase of decoding can start, using the last pair of RS decoders. An evaluation of the performances of this parity sharing RS decoder can be done starting from table II. In this table the characteristic of a 8-bits RS decoder, used for comparison, and the two 4-bits RS decoders used as building blocks of the parity sharing RS decoder are reported.

Symbol Width	8	4	4
k	32	13	6
n	36	15	10
Processing Delay	35	15	35
Latency	78	37	52
Area (Slices)	638	231	317
Area (Block RAM)	2	0	0
Max. Frequency	136MHz	203MHz	203MHz
Xilinx Part	XC2VP2-6	XC2VP2-6	XC2VP2-6

TABLE II
DECODERS

The main parameters to be taken into account are the area occupation and the symbol throughput. The area of the parity sharing decoder can be evaluated starting from the data reported in table II. The total number of slices for the chosen example, composed by four RS(15,13) decoders and one RS(10,6) decoder is about 1241 slices. The area is doubled with respect to the standard RS decoder, even if the single RS decoders blocks of the overall parity sharing decoder are quite small. This amount of area is due also to the choice of performing the third step of the decoding algorithm using a second pair of RS(N_1,K_1) decoders. The reuse of the same decoders can be taken into account to reduce the area occupation of the decoder, but increase the overall processing delay. It must be noticed that the processing delay of the parity sharing decoder depends on the number of RS(N_1,K_1) codeword that cannot be corrected during the first step of the algorithm, that must be reprocessed during the third step. This event depends on the M, N_1,K_1 parameters of the code and on the erasures and error rates, and therefore a detailed reliability analysis can help to choice between the use of the structure presented in Fig. 6, and a smaller structure that exploits the reuse of the first pair of decoders. To compare the symbol throughput of the RS decoder with the parity sharing one, we notice that the Maximum Clock Frequency available for the 4-bits RS decoders is quite higher than the one of the 8-bits RS decoders. To allow the continuous streaming of the first step of the parity sharing algorithm the following assumption must be verified:

$$(N_1 - K_1)(N_1 - K_1 + 4) + 3 \leq N_1$$

This assumption can be easily satisfied by limiting the number of check symbols of the first decoder. For the second pairs of decoders we can notice that the continuous streaming is not necessary because the intermediate data are stored in the RAM block. The only condition that must be satisfied is that the $N_1 - M$ column RS(N_2,K_2) codewords are decoded before than a new parity sharing codeword is processed. The processing delay for processing a RS(N_2,K_2) codeword is $(N_2 - K_2)(N_2 - K_2 + 4) + 3$ therefore the condition can be expressed as:

$$(N_1 - M)((N_2 - K_2)(N_2 - K_2 + 4) + 3) \leq K_2 M + (N_1 - M)(N_2 - K_2)$$

In the given example the condition become $78 < 86$. The continuous streaming of the third step is guaranteed from the continuous streaming of the first decoder. The symbol throughput can therefore be evaluated as a symbol each clock cycle, and in the given example is about 1.62 Gbits/sec. The latency is a lack of the parity sharing decoder. In fact, the three step for the decoding are performed in serial mode, therefore the latency is the sum of the latencies of the 4-bits RS decoders. In particular, for the chosen example we obtain a latency of 126 clock cycles, compared with a latency of 78 clock cycles of the standard RS decoder. If the systems are clocked at their maximum operating frequency we obtain a latency of 617 ns for the parity sharing decoder compared with a latency of 569 ns for the RS decoder.

V. CONCLUSIONS

The performance of block codes improves with codeword length. For RS codes, the length is constrained by the size of the field over which the code is defined. The computation and memory required to carry out Gal

arithmetic increases with field size and therefore, decoding a long RS code can be impractical. In particular for an FGPA, the implementation of small Galois Field can be very efficient compared with larger ones. Parity-sharing codes can be used to increase the performances of the hardware implementation of the codecs, maintaining the same data rate and a comparable error correction capability than traditional RS codes. In this paper the hardware implementation of parity sharing codecs is provided, the area and delay comparison of these structures with traditional RS codecs are carried out and the main gains in terms of performances are underlined.

REFERENCES

[1] O. Collins, 'Exploiting the cannibalistic traits of Reed-Solomon codes', Communications, IEEE Transactions on, Volume 43, Issue 11, Nov. 1995 Page(s):2696 - 2703
[2] M.K. Cheng, P.H. Siegel,'List-decoding of parity-sharing Reed-Solomon codes in magnetic recording systems', Communications, 2004 IEEE International Conference on Volume 2, 20-24 June 2004 Page(s):640 - 644 Vol.2
[3] R. E. Blahut, 'Theory and Practice of Error Control Codes', Addison-Wesley Publishing Company, 1983
[4] Xilinx Logic Core Reed-Solomon Encoder v5.0, Data sheet
[5] Xilinx Logic Core Reed-Solomon Decoder v5.0, Data sheet

Proceedings of the 2005 20th IEEE International Symposium on Defect and Fault Tolerance in VLSI Systems (DFT'05)
0-7695-2464-8/05 $20.00 © 2005 IEEE