Concurrent Error Detection in Reed Solomon Decoders

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

***Abstract*— Reed Solomon codes are widely used to identify and correct data errors in transmission and storage systems. When Reed Solomon (RS) codes are used for high reliable systems, the designer should take into account also for the occurrence of faults in the encoder and decoder blocks. In this paper a method to obtain a self-checking RS decoder is presented and different architectures for its implementation based on concurrent error detection are provided. The proposed method can be used for a wide range of different decoder algorithms with no intervention on the decoder architecture.**

I. INTRODUCTION

Error Correction Codes (ECC) are used in different applications, such as for example, to protect data transmitted over a noisy channel or to obtain high reliable data storage systems. Exploiting suitable redundancies these codes are able to detect and/or to correct errors in the binary representation of the data. The encoder take as input a certain amount of data, forming the dataword and provides as output a stream of bits forming a codeword. The codeword is composed by the information contained in the dataword plus some redundancies used by the decoder to check the correctness of the received data and correct the corrupted data. 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 occurs during the the codeword transmission. Therefore great attention must be paid to detect and recover faults in the encoding and decoding circuitry. These faults can be generated by different reasons such as technological process fails, aging of the electronic devices or by phenomena related to the scaling of the elementary electronic devices generating a greater susceptibility to the external environment (such as for example radiation effects at sea level). Moreover, ECC are widely used in space applications for the design of space-borne mass memories [1] and for the transmission of the collected data to the earth stations. These applications require high reliability, and the related systems must be tolerant to the effects induced by mechanical and thermal stresses and, especially, radiation related Single Event Upset (SEU) phenomena. Nowadays, the most used error correcting codes are the Reed-Solomon 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 operation, performed modulo 2, and the XOR operation between the bits representing the field elements. In [2], [3] the authors proposed to exploit this relationship to detect faults occurring in the encoder, achieving the self-checking property for the arithmetic blocks used in the encoder implementation. In [4], [5] a method to obtain Concurrent Error Detection (CED) circuits for finite field multipliers and inverters has been proposed. Since the Reed-Solomon decoder is based on GF(2^m) addition, multiplication and inversion, also the self checking decoder could be designed by using the CED implementations of these arithmetic blocks. Moreover in [6] a self-checking algorithm for solving the key equation (that is only a part of the overall decoding algorithm) has been introduced. Exploiting the algorithm proposed in [6] and substituting the elementary operations with the corresponding CED implementation for the other parts of the decoding algorithm a self-checking decoder can been implemented. This approach presents the following drawbacks:

1) The internal structure of the decoder must be modified by substituting the elementary operations with the corresponding CED ones. Therefore the decoder performances in terms of maximum operating frequency, area occupation and power consumption can be very different with respect to the non self-checking implementation.
2) The self-checking implementation is strongly dependent from the chosen decoder architecture (e.g. Berlekamp-Massey algorithm [7] or modified Euclidean algorithm [8]).
3) A good knowledge of the finite field arithmetic

0-7803-9390-2/06/$20.00 ©2006 IEEE

ISCAS 2006

is essential for the implementation of GF(2^m) arithmetic blocks.

In this paper, differently from the above discussed approaches, the implementation of the self-checking RS decoder is based on a standard RS decoder (see IP vendors [9], [10] for example) and by adding suitable hardware blocks outside the standard decoder to check its functionality the self-checking implementation is obtained. In this way the proposed method can be directly used for a wide range of different decoder algorithms.

The paper is organized as follows: Section II gives a background of the Reed Solomon codes and describes the properties of the decoder with respect to a fault occurring inside it. In Section III the architecture of the proposed self-checking Reed Solomon decoder is presented and some evaluations in term of area and delay overhead are provided. Finally, conclusions are drawn in Section IV.

II. Reed Solomon Codes Background

In this section a short background on RS codes is outlined. In [11], [12] more information about finite fields and RS codes are provided. A RS(n,k) code is characterized by a codeword length of n symbols composed starting from a k symbols dataword. Symbols composing the dataword and the codeword are represented as elements of a GF(2^m) field, and therefore are bytes of m bits. The overall data word is treated as a polynomial $d(x)$ of degree k with coefficients in GF(2^m), while the codeword is a polynomial $c(x)$ of degree n with coefficients in GF(2^m).

A codeword is generated using a polynomial $g(x)$ named generator polynomial. All valid codewords are exactly divisible by the generator polynomial. The general form of the generator polynomial is:

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

where $2t = n - k$ and α is a primitive element of the field, i.e. $\forall \beta \in GF(2^m) - \{0\}\ \exists i \in \mathbb{N}\ |\alpha^i = \beta$.

The codeword of a RS(n,k) code can be constructed in two ways. Given a dataword $d(x)$ of k symbols the non systematic RS(n,k) code is the product $c(x) = d(x) \cdot g(x)$, while the systematic RS(n,k) code is obtained as:

$$c(x) = d(x) \cdot x^{n-k} - p(x) \quad (2)$$

$$p(x) = d(x) \cdot x^{n-k}\ mod\ g(x) \quad (3)$$

In this case $p(x)$ is polynomial with degree less than $n - k$ representing the parity symbols. We underline that in both cases the obtained dataword is exactly divisible by the generator polynomial $g(x)$. Now we define the Hamming distance of two polynomial $a(x)$ and $b(x)$ of degree n as the number of coefficients of the same degree that are different i.e. $H(a(x), b(x)) = \#\{i \leq n | a_i \neq b_i\}$, and the Hamming weight W(a(x)) as the number of non-zero coefficients of $a(x)$, i.e. $W(a(x)) = \#\{i \leq n | a_i \neq 0\}$. It is easy to prove that $H(a(x), b(x)) = W(a(x) - b(x))$. In a RS($n$,$k$) code the Hamming distance between two codewords is $n - k$. After the transmission of a noisy channel the decoder receive as input a polynomial $\overline{c(x)} = c(x) + e(x)$, where $e(x)$ is the error polynomial. The Reed-Solomon decoder will identify the position and magnitude of up to t errors and it is able to correct them. In other words the decoder is able to identify the $e(x)$ polynomial if the Hamming weight $W(e(x))$ is not greater than t. The decoding algorithm provides as output the codeword that is the only codeword having an Hamming distance not greater than t from the received polynomial $\overline{c(x)}$. If the received polynomial $\overline{c(x)}$ contains more than t errors the decoder can provide as output a codeword with Hamming distance non greater that t, and a miscorrection is occurred. Therefore the correct behavior of a decoder can be identified by two (mean) main properties of the fault free decoder:

Property 1: The output of the decoder is always a codeword.

Property 2: The Hamming weight of the error polynomial is not greater than t.

If a fault occurs inside the decoder the observation outlined above are able to detect the occurrence of the fault. When the fault is activated, i.e. the output is different from the correct one due to the presence of the faults two cases can occur. The first one is that the decoder gives as output a non codeword, and this case can be detected by property 1. This is the most probable case because the decoder computes the error polynomial and obtains the output codeword by calculating $c(x) = \overline{c(x)} + e(x)$. However, even if the output of the faulty decoder is a wrong codeword the detection of this fault is easily performed by evaluating the Hamming weight of the error polynomial if it is provided by the decoder or evaluating the Hamming distance between the received inputs and the provided output. Therefore if one of the two properties is not respected a fault inside the decoder is detected, while if all the observations are satisfied we can detect that not faults are activated inside the decoder. We underline that this approach is completely independent by the assumed fault set and it is based only on the assumption that

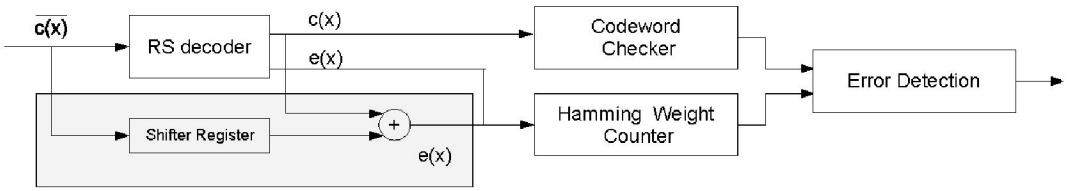

Fig. 1. CED scheme of the RS decoder

the fault free behavior of the decoder provides always a codeword as output. This assumption is valid for a wide range of decoder architectures even if some decoders are able to perform a miscorrection detection for some received polynomials with more than t errors.

III. Concurrent error detection scheme of the RS decoder

In Fig.1 a general scheme of the CED implementation of the RS decoder is shown. Its main blocks are:

- RS decoder, i.e the block that must be checked.
- An optional error polynomial recover (the shaded block shown in Fig. 1) that is needed if the RS decoder do not provide as output the error polynomial coefficients.
- Hamming weight counter, that checks the number of coefficients of the error polynomial that differs from zero.
- Codeword checker, that checks if the output data of the RS decoder forms a correct codeword.
- Error detection block that take as inputs the responses of the Hamming weight counter and of the codeword checker and provides an output signaling if a fault inside the RS decoder has been detected.

The RS decoder can be considered as a black box performing an algorithm for the error detection and correction of the input data (the coefficients of the polynomial $\overline{c(x)}$). We define L as the latency of the decoder i.e. the number of clock cycles from a symbol being sampled at the input, to the corrected version of that symbol appearing as output, and we suppose that the latency is fixed for the chosen decoder architecture. This hypothesis is not mandatory in order to apply the presented method but it is used only to simplify the proposed schemes. Many RS decoders provide as additional outputs the error polynomial (e.g. see[9], [10]) or the original input data delayed of L clock cycles.

If no additional outputs are provided we need to use the error polynomial recover block that is composed by a shifter register of length L and by a GF(2^m) adder that is obtained as a bitwise XOR of the coefficients of $c(x)$ and $\overline{c(x)}$. If only the delayed original input data are provided, the error polynomial recover block can be implemented by using only the GF(2^m) adder.

The Hamming weight counter is composed by:

1) A comparator that indicates (at each clock cycle) if the $e(x)$ coefficients are zero.
2) A counter that take into account the number of non zero coefficients.
3) A comparator between this number and t that is the maximum allowed number of non zero elements.

The codeword checker block checks if the reconstructed $c(x)$ is a codeword, i.e. if it is exactly divisible for the generator polynomial $g(x)$. Two implementations of this block can be used.

Implementation 1: It is based on computing the remainder of the polynomial division between $c(x)$ and $g(x)$. If all the coefficients of the remainder polynomial are zero then the polynomial $c(x)$ is a correct codeword. The remainder of the division for $g(x)$ is exactly the function of the systematic RS encoder. In fact a systematic RS encoder provides as check symbols the remainder of the division for $g(x)$. Therefore we can use a systematic RS encoder with the same $g(x)$ polynomial of the decoder to check the codeword correctness. We outline that if in the overall telecommunication system we use a systematic RS code we can detect faults in the decoder ignoring either the $g(x)$ polynomial used to create the codeword and also ignoring the way in which the operation in GF(2^m) are performed. We only need to reuse the same RS encoder used to create the codeword for the computation of the remainder of the polynomial $c(x)$ obtained from the decoder. The drawback of this implementation is the additional latency introduced by the RS encoder, that usually is $n-k$ clock cycles. This latency must be considered by the error detection block that waits $n-k$ clock cycles to check the two properties defined in the previous section. The area occupation of the RS encoder is smaller than the area occupation of the decoder (see e.g. [12] and [13]), therefore the overhead introduced by this block is evaluated to be about 15% of the decoder area.

Implementation 2: The codeword checker block is based on the so-called syndrome calculation. This operation is the first operation performed inside the decoder, therefore conceptually this approach implies a partial duplication of the RS decoder and implies the knowledge of the used Galois field and the roots of the generator polynomial $g(x)$. For the decoder, the syndrome calculation consists in the evaluation of the received polynomial $\overline{c(x)}$ for the values of x in the set A, with $A = \{\alpha^{i+j} | 0 \leq j \leq 2t\}$, i.e. A is the set of the roots of $g(x)$. The received polynomial $\overline{c(x)}$ is exactly divisible for $g(x)$ if and only if is exactly divisible for all the monomials $(x - \alpha^{i+j})$, if α is a root of $g(x)$. The polynomial is divisible $(x - \alpha^{i+j})$ if $\overline{c(\alpha^{i+j})}$ is zero. Therefore, the received polynomial is a codeword if and only if all the computed syndromes are zero. In Fig. 2 a block computing one of the $2t$ syndromes is presented. It is basically composed by a GF(2^m) constant multiplier, an adder and a m-bit register. The output of this block is the j-th syndrome and it is valid one clock cycle later the computation of the last coefficient of the polynomial. It must be noticed that the area occupation of the syndrome calculation block is equivalent to the encoder area occupation. In fact, in both cases we need $n - k$ blocks composed by an adder, a constant multiplier and a m-bit register. The difference between the two choices is the latency of the codeword checker block. The error detection block must take as inputs the responses of the the Hamming weight counter and of the codeword checker and its implementation depends from the chosen implementation of the codeword checker. If we have as inputs the remainder of the division by $g(x)$ this block must delay the response of the Hamming weight counter for $n-k$ clock cycles and checks if all the coefficients of the remainder polynomial are zero. On the other hand, if we use the syndromes calculation block the inputs are the computed syndromes and it check if all the received symbols are zero.

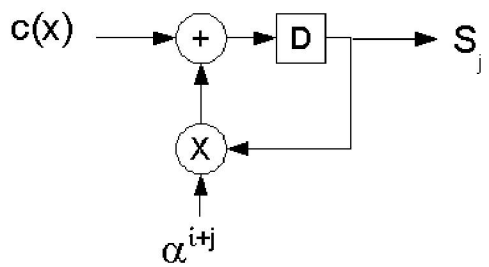

Fig. 2. Syndrome calculation block

IV. Conclusions

In this paper an innovative self-checking Reed Solomon decoder architecture is described. Two main properties of the behavior of the fault free decoder are identified and used to detect if a fault inside the decoder is activated. The proposed method can be used for a wide range of decoder algorithm and it is independent from fault set assumptions, and therefore by the chosen implementation technology. Some concurrent error detection schemes are explained in the paper and some evaluations in term of area overhead are provided. Our method is non intrusive, i.e. the decoder architecture is not modified and therefore the performances of the decoder in terms of maximum operating frequency, area occupation and power consumption are the same of the non self-checking implementation. Moreover, the main properties of the decoder identified in the paper permits to obtain a self checking architecture with only few knowledge of the arithmetic of finite fields.

References

[1] G.C. Cardarilli, A. Leandri, P. Marinucci, M. Ottavi, S. Pontarelli, M. Re, A. Salsano, Design of a fault tolerant solid state mass memory, Reliability, IEEE Transactions on Volume 52, Issue 4, Dec. 2003 pp:476 - 491

[2] G.C. Cardarilli, S. Pontarelli , M. Re, A. Salsano, "Design of a Self Checking Reed Solomon Encoder", Proceedings of the 11th IEEE International On-Line Testing Symposium (IOLTS 2005), July 2005.

[3] G.C. Cardarilli, S. Pontarelli , M. Re, A. Salsano, "A Self Checking Reed Solomon Encoder: Design and Analysis", IEEE International Symposium on Defect and Fault Tolerance in VLSI Systems, DFT 2005, Monterey, CA, USA, October 2005.

[4] Gossel, M.; Fenn, S.; Taylor, D., "On-line error detection for finite field multipliers", Defect and Fault Tolerance in VLSI Systems, 1997. Proceedings., 1997 IEEE International Symposium on 20-22 Oct. 1997 pp:307 - 311

[5] Yu-Chun Chuang; Cheng-Wen Wu; "On-line error detection schemes for a systolic finite-field inverter", Proceedings of the Seventh Asian Test Symposium, 1998. ATS '98, pp:301 - 305

[6] I. M. Boyarinov, "Self-Checking Algorithm of Solving the Key Equation", Proceedings of IEEE International Symposium on Information Theory, 1998.

[7] Sarwate, D.V.; Shanbhag, N.R.; "High-speed architectures for Reed-Solomon decoders" Very Large Scale Integration (VLSI) Systems, IEEE Transactions on, Vol. 9, Issue 5, Oct. 2001 Page(s):641 - 655

[8] Hanho Lee, "High-speed VLSI architecture for parallel Reed-Solomon decoder", Very Large Scale Integration (VLSI) Systems, IEEE Transactions on Vol. 11, Issue 2, April 2003 pp:288 - 294

[9] Altera Reed-Solomon compiler User Guide 3.3.3

[10] Xilinx Logicore Reed-Solomon Decoder v5.1

[11] Lidl, R. and Niederreiter, H. "Introduction to Finite Fields and Their Applications", rev. ed. Cambridge, England: Cambridge University Press, 1994.

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

[13] Xilinx Logicore Reed-Solomon Encoder v5.1