Forward Backward Syndrome Computation: A Reduced Complexity CRC Code Decoder

Titouan Gendron, Emmanuel Boutillon, Charbel Abdel Nour, David Gnaedig

Forward Backward Syndrome Computation: A Reduced Complexity CRC Code Decoder

Titouan Gendron, Emmanuel Boutillon, Charbel Abdel Nour and David Gnaedig

***Abstract*—This paper proposes a simplified method for correcting a single-bit error event in a message protected by a CRC code thanks to a new decoding method called Forward-Backward Syndrome Computation. The two key ideas of the paper are to show that: 1) syndrome computation can be performed in any direction (Forward and Backward), 2) consequently, it is possible to create a partition of the received message in a set of disjoint syndromes, thus allowing a direct identification of the error position. Evaluation in terms of number of 2-input binary logic functions shows a complexity reduction by a factor of up to 3 compared to conventional methods.**

***Index Terms*—CRC code, error correction, single-bit error, syndrome computation**

I. Introduction

CYCLIC Redundancy Check (CRC) codes are widely used in communication systems for their error detection capabilities. However, CRC codes can also be used to correct few errors. It is known since the sixties [1] that a single-bit error of a CRC code can be corrected with a complexity proportional to the code length N. Briefly speaking, the method consists of comparing the syndrome with a pre-computed table containing all the possible syndromes associated to a single error event.

More schemes have been introduced since [1] to decode CRC codes. These methods can be split into code agnostic methods (like the GRAND algorithm [2] and its numerous variations using soft information) or code aware methods. The latter category can be split into two sub-categories: 1) Hard decision decoding algorithms that require a limited complexity. They are used to correct only single-bit errors or double-bit errors due to the exponential increase in complexity with the number of corrected errors. Similar to [3], [4], most methods are based on variations of the table method described in [1]. Recently, [5] proposed a "table-free method" using specific properties of the CRC code to decode one or two errors. 2) Soft decision decoding algorithms that can correct more than two errors thanks to the use of soft input information at the cost of a larger complexity, such as in [6]–[9].

This work focuses on the proposal of a code-aware hard decoding scheme to correct a single-bit error with minimum complexity and latency. It proposes a new decoding method called Forward-Backward Syndrome Computation (FB-SC) that reduces, by a factor of up to 3, the complexity of the classical single-bit error decoder. The first key contribution of this work is to show that syndrome computation can be performed in any direction (Forward or Backward). The "Forward" direction denotes the classical way to compute the syndrome, while the "Backward" direction denotes the bit-reversed order. Note that the proposed algorithm is completely different from the well known BCJR algorithm, used to decode a convolutional code and also referred to as the Forward-Backward algorithm. The only commonality resides in the direction of the applied processing schedule.

The other key contribution of this paper is the proof that the computation of a partial syndrome based on any set of M consecutive positions (M being the length of the redundancy part of a codeword) can be used to correct an error within these positions. This work can be seen as a specific and efficient implementation of the more general Information Set Decoding algorithm [10]. It is motivated by the industrial demand of an improved decoder for LTE Turbocode [11].

The rest of the paper is organized as follows. Section II reminds the basics of a CRC code and defines the notations. Section III introduces the proposed FB-SC decoding method. Section IV explains how to use the FB-SC method to correct a single error event, while section V concludes the paper.

II. CRC codes

Let K, N and R be the information length, the codeword length and code rate of the polynomial code, respectively. The polynomial code is defined by the generator polynomial $g(x) = x^M + g_{M-1}x^{M-1} + \dots g_1x^1 + 1$ of degree $M = N - K$. The message $\mathbf{u} = (u_i)_{i=0,1,\dots,K-1}$ of length K is expressed as a polynomial of degree $K-1$ noted $u(x) = u_{K-1}x^{K-1} + u_{K-1}x^{K-2} + \dots u_1x^1 + u_0x^0$. The polynomial redundancy $r(x)$ of degree $M-1$ is computed as the remainder of the polynomial division of $u(x)x^M$ by $g(x)$,

$$r(x) = u(x)x^M \mod g(x), \tag{1}$$

where $\mod$ is the modulo operator. From $r(x)$, the codeword $c(x)$ is generated[1] as

$$c(x) = u(x)x^M + r(x). \tag{2}$$

By construction, $c(x)$ is a multiple of the generator polynomial $g(x)$ and thus $c(x) = 0 \mod g(x)$. After transmission on a Binary Symmetric Channel (BSC) with a probability of error p, the received message is

$$y(x) = c(x) + e(x), \tag{3}$$

This work was granted by ANRT and Turboconcept, CIFRE PhD 2018.0921

Titouan Gendron and David Gnaëdig are with Turboconcept, Brest, France (e-mail: titouan.gendron@univ-ubs.fr; david.gnaedig@turboconcept.com).

Emmanuel Boutillon is with Lab-STICC (UMR 6285, CNRS), Université Bretagne Sud, Lorient, France (e-mail: emmanuel.boutillon@univ-ubs.fr).

C. Abdel Nour is with Lab-STICC (UMR 6285, CNRS), IMT-Atlantique, Brest, France (e-mail: charbel.abdelnour@imt-atlantique.fr).

[1]From $c(x)$, the binary encoded message is given as $\mathbf{c} = (u_{K-1}, u_{K-2}, \dots, u_0, r_{M-1}, r_{M-2}, \dots, r_0)$.

where $e(x)$ represents the error polynomial that contains non-zero coefficients at each error position. From (3), we get $s(x) = y(x) \mod g(x) = (c(x)+e(x)) \mod g(x)$ and thus, using the linearity of the modulo operation,

$$s(x) = e(x) \mod g(x). \tag{4}$$

The polynomial $s(x)$ is called the syndrome. It depends only on the error vector $e(x)$ and the generator polynomial $g(x)$ through (4). If $s(x)$ is equal to 0, then an error-free transmission is assumed (when the error sequence $e(x)$ is a codeword, an undetected error event occurs). When the syndrome $s(x)$ differs from zero, then $e(x)$ is a non-null polynomial and its value should be inferred from $s(x)$. In the sequel, the Hamming weight of a polynomial $h(x)$, i.e. its number of non-zero elements will be denoted $w(h)$.

III. The FB-SC correction method

This section first discusses the symmetric property of the CRC code, then explains how this symmetry can be used to correct errors. Symmetry refers to the ability to construct a codeword from a message $u(x)$ either in the "Forward" form $c_0(x) = u(x)x^M + r_0(x)$, or in the "Backward" form $c_1(x) = r_1(x)X^K + u(x)$. Note that $c_1(x)$ is a different codeword of the same CRC code and $r_1(x)$ can be processed from the reverse-ordered generator polynomial with a method described next.

A. Property of symmetric encoding

Lemma 1: Let $a(x)$ be a polynomial of degree n and $(a_n a_{n-1} \dots a_1 a_0)$ its associated binary representation. Then, a polynomial $a(x^{-1})x^n$ is a polynomial of degree n with a binary representation given in reversed order, i.e. $(a_0 a_1 \dots a_{n-1} a_n)$.

Proof The i^{th} monomial, $i = 0, 1, \dots, n$, of $a(x)$, i.e. $a_i x^i$, is associated to the $(n-i)^{th}$ monomial of $a(x^{-1})x^n$ since $a_i x^{-i} x^n = a_i x^{n-i}$ □

As $c(x)$ is a multiple of $g(x)$, a unique quotient $q(x)$ exists, such that

$$c(x) = q(x)g(x). \tag{5}$$

By replacing x by x^{-1} in (5) and by multiplying both sides of the equality by x^{N-1}, we obtain

$$c(x^{-1})x^{N-1} = q(x^{-1})g(x^{-1})x^{N-1}. \tag{6}$$

The x^{N-1} term on the right side of (6) can be decomposed into the product $x^{K-1}x^{N-K}$, or equivalently, $x^{K-1}x^M$ since $M = N - K$. Thus, (6) becomes

$$c(x^{-1})x^{N-1} = (q(x^{-1})x^{K-1})(g(x^{-1})x^M). \tag{7}$$

From (7), we can see that the reversed-order polynomial $\bar{c}(x) = c(x^{-1})x^{N-1}$ is a multiple of the reversed-order polynomial $\bar{g}(x) = g(x^{-1})x^M$. In other words, $\bar{\mathbf{c}} = (c_0, c_1, \dots, c_{N-2}, c_{N-1})$ is a codeword of the code generated by the polynomial $\bar{\mathbf{g}} = (g_0, g_1, \dots, g_{M-1}, g_M)$. Therefore, it becomes possible to revisit the decoding process in a symmetrical way by reversing the order of all polynomials. If we call "Forward" the natural way to compute a syndrome (the natural bit order), the symmetrical way can then be defined as "Backward" (the bit-reversed order).

Let us define the syndrome $\bar{s}(x)$ as $\bar{s}(x) = y(x^{-1})x^{N-1} \mod \bar{g}(x)$. This equation gives $y(x^{-1})x^{N-1} + \bar{s}(x) = 0 \mod \bar{g}(x)$ and $\bar{q}(x)$ exists such that

$$y(x^{-1})x^{N-1} + \bar{s}(x) = \bar{q}(x)\bar{g}(x). \tag{8}$$

Replacing in (8) x by x^{-1} and multiplying both terms by x^{N-1} reverts the equation to the natural order. For example, the term $y(x^{-1})x^{N-1}$ is switched back using $y((x^{-1})^{-1})(x^{-1})^{N-1}x^{N-1} = y(x))$. The whole equation gives

$$y(x) + \bar{s}(x^{-1})x^{N-1} = q'(x)g(x), \tag{9}$$

with $q'(x) = \bar{q}(x^{-1})x^{N-1}$. In other words, $y(x) + \bar{s}(x^{-1})x^{N-1}$ is a codeword of the code generated by the polynomial $g(x)$. The polynomial $\bar{s}(x^{-1})x^{N-1}$ is of degree $N-1$, but its K lowest degree coefficients are all equal to zeros.

In the next section, we combine forward and backward remainders to compute a middle remainder and associated middle syndromes.

B. Computation of a middle syndrome

We propose a generalization of the capability of finding a codeword by correcting errors within any adjacent subset of M bits starting from index l and ending at index $l+M-1$, with $l = 0, 1, \dots, N-M-1$. More precisely, we present a method that provides a polynomial codeword from a vector $y(x)$ of size N bits by only modifying the subset of coefficients between positions l and $l+M-1$

To do so, let us decompose $y(x)$ into three vectors as follows (see Fig. 1 and Fig. 2)

$$y(x) = y_l^f(x)x^{l+M} + y_l^m(x)x^l + y_l^b(x) \tag{10}$$

with $y_l^f(x)$ corresponding to the polynomial constructed with the $N-M-l$ highest degree coefficients of $y(x)$ (exponent f for "Forward coefficients", this name is justified later by the fact that they are processed in the forward direction), $y_l^m(x)$ the polynomial constructed with the M coefficients of $y(x)$ between x^l and x^{l+M-1} (exponent m corresponding to "middle coefficients") and $y_l^b(x)$ the polynomial corresponding to the l lowest coefficients of $y(x)$ (exponent b for "Backward coefficients"). These vectors are defined as

$$\begin{aligned} y_l^f(x) &= \sum_{i=l+M}^{N-1} y_i x^{i-M-l}, \\ y_l^m(x) = \sum_{i=l}^{l+M-1} y_i x^{i-l} \quad &\text{and} \quad y_l^b(x) = \sum_{i=0}^{l-1} y_i x^i. \end{aligned} \tag{11}$$

We denote by r_l^f the partial forward remainder such that

$$r_l^f(x) = y_l^f(x)x^M \mod g(x) \tag{12}$$

Multiplying both terms of (12) by x^l and regrouping the terms on the left side gives

$$y_l^f(x)x^{M+l} + r_l^f(x)x^l = 0 \mod g(x). \tag{13}$$

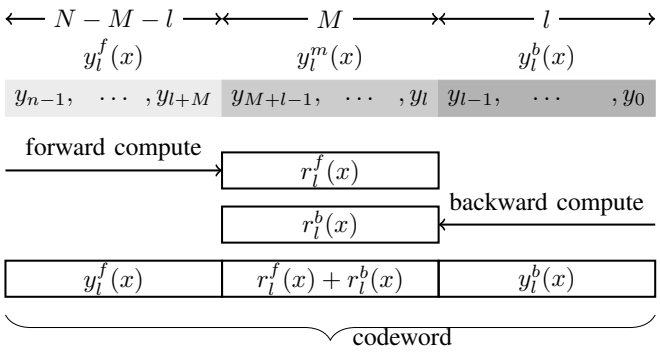

Fig. 1: Illustration of the FB-SC principle.

For a bit-reversed order processing, we can define the partial backward remainder $\bar{r}_l^b(x)$ from the reversed-order polynomial $\bar{y}_l^b(x) = y_l^b(x^{-1})x^l$ multiplied by x^M and the reversed-order key $\bar{g}(x)$ as

$$\bar{r}_l^b(x) = \bar{y}_l^b(x)x^M \mod \bar{g}(x). \tag{14}$$

Therefore, $\bar{y}_l^b(x)x^M + \bar{r}_l^b(x) = 0 \mod g(x)$, and thus, there exists $\bar{q}(x)$ such that

$$\bar{y}_l^b(x)x^M + \bar{r}_l^b(x) = \bar{q}(x)\bar{g}(x). \tag{15}$$

Similarly to the transformation between (8) and (9), replacing x by x^{-1} in (15) and multiplying both terms by x^{M+l} gives

$$y_l^b(x) + r_l^b(x)x^l = 0 \mod g(x), \tag{16}$$

where $r_l^b(x) = \bar{r}_l^b(x^{-1})x^M$ denotes the reversed-order polynomial of $\bar{r}_l^b(x)$.

By adding the two terms $r_l^f(x)$ and $r_l^b(x)$, we can define the middle (or generalized) remainder $r_l^m(x)$ as

$$r_l^m(x) = r_l^f(x) + r_l^b(x) \tag{17}$$

and the middle syndrome $s_l^m(x)$ at the level of the bit l within the frame, as

$$s_l^m(x) = y_l^m(x) + r_l^m(x). \tag{18}$$

The corresponding process is illustrated in Fig. 1. The sum of the polynomials in (13) and (16) verifies

$$y_l^f(x)x^{M+l} + r_l^m(x)x^l + y_l^b(x) = 0 \mod g(x), \tag{19}$$

or equivalently, is also a codeword of the polynomial code.

Note that it can be demonstrated that, if $y(x)$ is a codeword, then for all l, $s_l^m(x) = 0$, and also its reciprocal property, i.e. if index l exists such that $s_l^m(x) = 0$, then $y(x)$ is a codeword.

In the following, a method to compute the middle syndrome is presented.

C. Recursive computation of $r_l^f(x)$ and $r_l^b(x)$

Following the definition of $y_l^f(x)$ in (11), we can write

$$y_l^f(x) = y_{l+1}^f(x)x + y_{l+M}. \tag{20}$$

Starting from (12), for $l = K-1$ down to 0 we have

$$r_l^f(x) = (y_{l+1}^f(x)x + y_{l+M})x^M \mod g(x) \tag{21}$$
$$= (r_{l+1}^f(x)x + y_{l+M}x^M) \mod g(x) \tag{22}$$

with $r_{l=K}^f(x) = 0$. Note that (22) is the mathematical expression used to compute the final redundancy $r(x) = r_0^f(x)$ of the CRC encoding process when $y(x) = u(x)x^M$.

The computation of $\bar{r}_l^b$ from $l = 0$ to $K-1$ can be performed by applying a symmetrical encoding approach following

$$\bar{r}_l^b(x) = (r_{l-1}^b(x)x + y_l x^M) \mod \bar{g}(x), \tag{23}$$

with $r_{l=-1}^b(x) = 0$.

Without loss of generality, we assume that N is a multiple of M, i.e. $N = \gamma M$. If it is not the case, $y(x)$ can be padded with dummy null coefficients until its size matches a multiple of M. It is then possible to partition $y(x)$ into γ segments of size M each, and to compute the γ middle syndromes associated to each segment $s_{kM}^m(x) = r_{kM}^m(x) + y_{kM}^m(x)$, with $k = 0, 1, \ldots, \gamma - 1$. To do so, the partial remainders $r_{kM}^f(x)$ from (22) and $r_{kM}^b(x)$ from (23) computed for $k = 0, 1, \ldots, \gamma - 1$ have to be saved periodically every M cycles. Finally, after reordering, the middle remainder denoted by $r_{kM}^m(x)$ is computed (see (17)) followed by the computation of the middle syndromes $s_{kM}^m(x)$ (see (18)), with $k = 0, 1, \ldots, \gamma - 1$.

The next section shows how to simplify the correction of a single error event using the partitions of FB syndromes.

IV. Correction of the single-bit error event

In this section, we first present the standard method to correct a single-bit error event and we evaluate its complexity. Then, we show how to benefit from the set of FB syndromes associated to the segments of $y(x)$ to reduce the complexity of the correction by a factor M.

A. Classical single-bit error correction method

In case of a single-bit error at position a, i.e. $e(x) = x^a$, then, the corresponding syndrome is $s(x) = s_a^{(1)}(x)$, with

$$s_a^{(1)}(x) = x^a \mod g(x), \tag{24}$$

where the superscript $^{(1)}$ indicates a syndrome for an error event of Hamming weight of 1 (single-bit error) and the subscript a indicates the position of the error.

The set of N syndromes $\mathcal{S}^{(1)} = \{s_i^{(1)}(x)\}_{i=0,1,\ldots,N-1}$ can be precomputed to correct every single-bit error event with a complexity in $\mathcal{O}(N)$. To this end, $s(x)$ is compared to the values of $\mathcal{S}^{(1)}$ until finding the value a verifying $s(x) = s_a^{(1)}(x)$. Hence, the resulting codeword is obtained from $y(x)$ by flipping the bit at the a^{th} position. When $s(x) \neq 0$ and $s(x) \notin \mathcal{S}^{(1)}$, then at least 2 errors occurred and the decoding process fails. Note that this method is applicable only if all elements of $\mathcal{S}^{(1)}$ are distinct. When $g(x)$ is irreducible over the set of polynomials with binary coefficients GF(2)[x], this hypothesis is fulfilled for $N < 2^M$. This method is called the table method [3]. Recently, in [5], another algorithm called "Table-Free Multiple Bit-Error Correction" (TFMBEC)

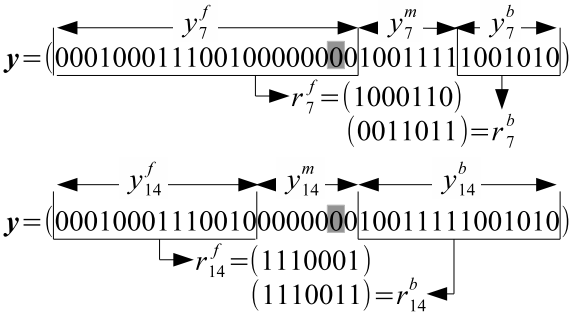

Fig. 2: Computation example of middle remainders

was proposed. However, the proposed algorithm involves data dependent iterations and, in the worst case (especially for a hardware implementation since constraints are set during execution time), it has a higher complexity than the table method (see section IV.B).

B. FB-SC correction method

Let us assume that there is only one error at position a, with

$$a = k_a M + \delta_a \tag{25}$$

and $0 \leq \delta_a < M$. In others words, the error belongs to the k_ath segment. Then, using the same arguments as the ones used to derive (4), the middle syndromes depend only on the error pattern $e(x)$. Let us focus on the middle syndrome $s^m_{k_a M}$. Since $e(x) = x^a$, we have $e^f_{k_a M}(x) = 0$, $e^m_{k_a M}(x) = x^{\delta_a}$, and $e^b_{k_a M}(x) = 0$. Thus, the partial Forward and Backward remainders linked to $e(x)$ are both null, giving $s^m_{k_a M} = x^{\delta_a}$, i.e. a weight-1 polynomial. The value δ_a identifies the index $a = k_a M + \delta_a$ of the bit to be flipped in $y(x)$ to obtain a codeword. The following toy example illustrates the proposed method. Let us assume $M = 7$, the generator polynomial $g(x) = x^7 + x^3 + 1$ and the length $K = 28$ message $\mathbf{u} = (0001000111001000000101001111)$. Based on the recursive equation (22), the obtained codeword $\mathbf{c}$ is given as $\mathbf{c} = (0001000111001000000101001111 1001010)$. Let us assume an error at position $a = 15$, the noisy received message becomes $\mathbf{y} = (0001000111001000000\mathbf{0}010011111001010)$. The index $a = 15$ can be decomposed following (25) such that $k_a = 2$ and $\delta_a = 1$. Fig. 2 shows graphically the computation of the 4 remainders (r^f_{14}, r^b_{14}) and (r^f_7, r^b_7). From this figure, we can deduce that $r^m_{14} = r^f_{14} + r^b_{14} = (0000010)$, and thus, that $s^m_{14} = y^m_{14} + r^m_{14} = (0000000) + (0000010) = (0000010)$, i.e. a weight-1 middle syndrome. Similarly, we get $r^m_7 = (1011101)$ and $s^m_7 = (1011101)$, i.e. weight-5 syndromes. The other middle syndromes for this example are $s^m_{28}(x) = (1001111)$, $s^m_{21}(x) = (0010011)$ and $s^m_0(x) = (0001011)$. As predicted, the only weight-1 syndrome is s^m_{14}, i.e. the one that is associated with the position of the error.

C. Complexity comparison

In this section, we compare the complexities of the classical and the proposed FB-SC methods. Both methods apply a 2-phase approach consisting of syndrome determination followed by error mitigation.

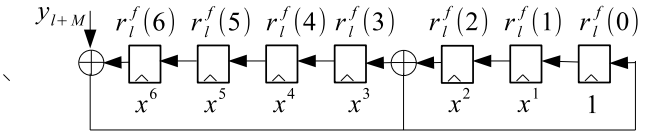

Fig. 3: Structure of LFSR for forward syndrome computation

The recursive computation of $r^f_l(x)$ is given in (22). Starting from $r^f_K(x) = 0$, the recursion $r^f_l(x) = (r^f_{l+1}(x)x + y_{l+M}x^M) \bmod g(x)$ is performed from $l = K - 1$ down to 0. This is a well known Linear Feedback Shift Register (LFSR) structure, similar to the one represented in Fig. 3 for $M = 7$, $g(x) = x^7 + x^3 + 1$. In the general case, the number of XOR operations to perform the full recursion is given as $N(w(g) - 1)$, where $w(g)$ denotes the number of non-zero values of the generator polynomial g. For the FB-SC method, both forward and backward recursions need to be performed, which doubles this complexity. Moreover, N additional XOR operations are required to compute the middle syndrome (*i.e.* $\gamma = N/M$ computations of (17), each requiring M XOR operations). The overall complexity for syndrome computation of the FB-SC method becomes $2N \times (w(g) - 1) + N$ XOR operations. While the FB-SC method requires a larger number of computations for the first phase of the algorithm, its complexity is considerably reduced when applying the second phase of single-bit error correction.

The complexity of the classical method is given as the cost to compare the N syndromes of the set $\mathcal{S}^{(1)}$ with the actual syndrome $s(x)$ of the received codeword. This is motivated by the fact that the comparison of two vectors of size M (*i.e.* the size of the syndrome) requires M XOR operations to sum them, then a M-entry NOR gate to test that the resulting vector is equal to the null vector. This latter operation requires a binary tree of M-2 OR gates followed by an NOR gate inverter. The complexity of this method can be approximated to $N \times M$ required XOR and OR operations.

Applied during the second phase, the FB-SC decoder checks if one of the middle syndromes has a Hamming weight of one. Several ways can be used to achieve this goal. We propose to derive an upper bound on the corresponding complexity. In order to simplify the notations, we omit the exponent m (denoting middle) as well as the subscript kM of the considered middle syndrome in the following description.

Let us thus consider the received middle section $y(x)$, the associated remainder $r(x)$ and syndrome $s(x) = y(x) + r(x)$, each of size M. If $s(x)$ has a weight 1, then the corrected version is equal to $\hat{c}(x) = y(x) + s(x)$. For a given index l in the range 0 to $M - 1$, let us define the Boolean variable $T_u(l)$ (subscript u for up) equals to one if the set $(s_{M-1}, s_{M-2}, \ldots, s_{l+1})$ contains at least one value 1, 0 otherwise. Similarly, we define $T_d(l)$ (subscript d for down) that equals 1 if the set $(s_l, s_{l-1}, \ldots, s_0)$ contains at least one value 1, 0 otherwise. By construction, $T_u(l) = T_u(l+1)$ OR s_l and $T_d(l) = T_d(l-1)$ OR s_l. For a given index j, if the 3 following conditions 1) $T_u(j+1) = 0$ (all syndrome values between $j+1$ and $M-1$ are equal to 0), 2) $T_d(j-1) = 0$ (all syndrome values between 0 and $j-1$ are equal to 0)

TABLE I: Complexity Comparison in Terms of 2-Input Binary Operations Between the Classical and The FB-SC Methods.

		Classical	FB-SC
General case	Syndrome	$Nw(g) - N$	$2Nw(g) - N$
	Correction	$2NM$	$5N$
	Total	$N(2M + w(g) - 1)$	$N(2w(g) + 4)$
LTE CRC	$g_a(x)$	61N	32N
	$g_b(x)$	53N	16N

and 3) $s_j = 1$ are met, then $s(x)$ is a weight-1 polynomial and the corrected bit is obtained by flipping the bit in the j^{th} position. If one of these 3 conditions is not fulfilled, then the bit should not be flipped. This operation can be implemented by the Slice Middle Syndrome Correction (SMSC) component given in Fig. 4.a). The serial association of M SMSC, as shown in Fig. 4.b), gives the Middle Syndrome Correction (MSC) components. MSC allows correcting a single-bit error in a middle syndrome. The initial conditions are $T_u(M) = 0$ and $T_d(0) = 0$. By construction, if the Hamming weight of the syndrome is larger than 1, or equal to 0, the MSC just leaves $y(x)$ unchanged. The MSC is thus used γ times during the decoding process, one for each length-M segment of the partition. Fig. 4.c shows the correction of the middle segments y_{14}^m and y_7^m given in the example of section IV.B.

The complexity per bit can be approximated by 5 binary operations (one XOR and 2 OR operations for the computation of $T_u(l)$ and $T_d(l)$ and 2 OR for the computation of $f(l) = T_u(l+1)$ OR $T_u(l-1)$ OR NOT(s_l), leading to an overall complexity of $5N$ two-inputs Boolean operations. Note that this estimate is slightly pessimistic. In fact, due to side effects, the first and the last SMSCs of the MSC require only 2 OR gates instead of 4. Table I compares the complexity of the two methods. Although one can argue about the difference in the relative complexity between the hardware of a XOR and an OR operation, the results are intended to provide a simple insight on how FB-SC reduces the decoding complexity compared to the classical method.

The optimized implementation of the TFMBEC algorithm [5] requires first $N(w(g) - 1)$ XOR operations for the syndrome computation. Then, the single-bit error correction requires N iterations. For each decoding iteration, a verification condition is performed using the Hamming weight of a length M vector. If the result is greater than 1, then $w(g)$ XOR operations are performed. By approximating the verification step to $4M$ gates operations, the worst case complexity is given as $N(2w(g) - 1 + 4M)$ gate operations. This number is greater than both the classical and the FB-SC methods (see Table I).

As an application example, Table I shows also the decoding complexity for the $M = 24$ CRC codes used in the LTE standard [12]. This latter specifies two CRC codes with generator polynomials $g_a(x)$ and $g_b(x)$ that contain 14 and 6 non-zero values, respectively. The FB-SC method reduces the complexity by a factor 2 for $g_a(x)$ and 3 for $g_b(x)$.

V. Conclusion

The paper presents a simplified method called FB-SC to detect and correct all single-bit error of a CRC code. The principle is to simplify the error detection and correction of a single-bit error by computing a partition of middle syndromes. In our future work, we will extend the FB-SC method to correct two-error patterns. Preliminary results confirm the appeal of the proposed method.

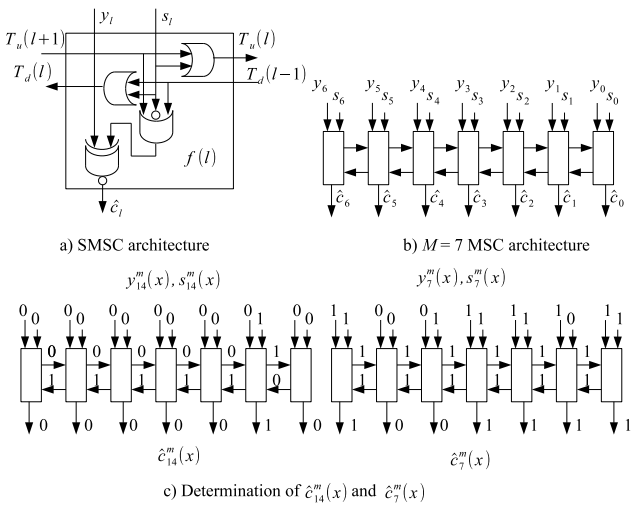

Fig. 4: Example of architecture for the correction of the 1 bit flip-error event

References

[1] W. W. Peterson and D. T. Brown, "Cyclic Codes for Error Detection," *Proceedings of the IRE*, vol. 49, no. 1, pp. 228–235, 1961.

[2] K. R. Duffy, J. Li, and M. Médard, "Capacity-Achieving Guessing Random Additive Noise Decoding," *IEEE Transactions on Information Theory*, vol. 65, no. 7, pp. 4023–4040, 2019.

[3] A. Habibizad Navin, S. H. Es-hagi, M. D. Yam, M. Hajiagapour, and M. Mirnia, "Data-oriented architecture for double and single bits error correction using Cycle Redundancy Code," in *Int. Conf. On Computer Design and Applications*, vol. 4, 2010, pp. V4–549–V4–552.

[4] S. Shukla and N. Bergmann, "Single bit error correction implementation in CRC-16 on FPGA," in *Proceedings. 2004 IEEE International Conference on Field- Programmable Technology (IEEE Cat. No.04EX921)*, 2004, pp. 319–322.

[5] V. Boussard, S. Coulombe, F.-X. Coudoux, and P. Corlay, "Table-Free Multiple Bit-Error Correction Using the CRC Syndrome," *IEEE Access*, vol. 8, pp. 102 357–102 372, 2020.

[6] E. Tsimbalo, X. Fafoutis, and R. J. Piechocki, "Crc error correction in iot applications," *IEEE Transactions on Industrial Informatics*, vol. 13, no. 1, pp. 361–369, 2017.

[7] T. Tonnellier, C. Leroux, B. Le Gal, B. Gadat, C. Jego, and N. Van Wambeke, "Lowering the Error Floor of Turbo Codes With CRC Verification," *IEEE Wireless Commun. Lett.*, vol. 5, no. 4, pp. 404–407, Aug 2016.

[8] M. Fossorier and S. Lin, "Soft decision decoding of Linear Block Codes based on Ordered Statistics," in *Proceedings of 1994 IEEE International Symposium on Information Theory*, 1994, pp. 395–.

[9] E. Tsimbalo, X. Fafoutis, and R. Piechocki, "Fix it, don't bin it! - CRC error correction in Bluetooth Low Energy," in *2015 IEEE 2nd World Forum on Internet of Things (WF-IoT)*, 2015, pp. 286–290.

[10] E. Prange, "The use of Information Sets in decoding Cyclic Codes," *IRE Transactions on Information Theory*, vol. 8, no. 5, pp. 5–9, 1962.

[11] T. Gendron, E. Boutillon, C. Abdel Nour, and D. Gnaedig, "Revisiting augmented decoding techniques for LTE Turbo Codes," in *2021 11th International Symposium on Topics in Coding (ISTC)*, 2021, pp. 1–5.

[12] 3GPP, "Evolved Universal Terrestrial Radio Access (E-UTRA); Radio Resource Control (RRC)," 3rd Generation Partnership Project (3GPP), Technical Specification (TS) 36.331, 04 2017, version 14.2.2.