A Reconfigurable FEC system based on Reed-Solomon codec for DVB and 802.16 network

LAMIA CHAARI, MOHAMED FOURATI, NOURI MASMOUDI, LOTFI KAMOUN
Electronic and Information Technologies Laboratory (L.E.T.I)
Sfax National Engineering School, 3038 SFAX TUNISIE
Lamia.chaari@tunet.tn
Mohamed.Fourati@isecs.rnu.tn

Abstract: This article proposes, a reconfigurable FEC system based on Reed-Solomon codec for DVB and WiMax networks. The proposed architecture implements various programmable primitive polynomials. A lot of VLSI implementations have been described in literature. This paper introduces a highly parametrical RS-coder-decoder on FPGAs. The implementation, written in a hardware description language (HDL), is based on an Berlekamp massey, Chain and Formey Algorithms. We have defined an advanced RS encoder-decoder architecture based on parameterization approach which is a key solution for software defined radio (SDR) systems. Our parameterization approach is used in order to implement on FPGA a generic RS coder-decoder for DVB and WiMax networks. IEEE Std. 802.16 specifies that the codec performs a variable number of check symbols in a codeword (ranges from 0 to 32, inclusive). The value of check symbols are specified for each burst profile by the MAC layer according to cross layer concept.

Keywords and phrases: Reedsolomon, Berlekamp massey, Chain, Formey, FPGA, implementation , VHDL, WiMax, DVB.

1 Introduction:

Reed-Solomon (RS) codes are described in a paper by Reed and Solomon in 1960[1]. RS are powerful error correcting codes that can be employed in a wide variety of digital communications systems from digital media to wireless communications and deep-space probes as well as in memory and storage systems. Reed-solomon codes are used to correct errors in many systems including:

- storage devices (compact disk, DVD, barcodes,etc….) [2,3],
- wireless and mobile communications (including cellular telephones, microwave links, etc…)[4, 5],
- digital satellite communications[6],
- digital television, digital video broadcasting (DVB)[7],
- high speed modem such as ADSL, Xdsl…[8]
- power line communications (PLC) [9]
- digital vestigial sideband (VSB) system [10]
- cable modem [11],

The speed and complexity of these systems necessitate designers and researchers to break a way from traditional architectures and design methodologies. A design combining software and hardware is flexible enough to make it optimal for 4th generation systems [12].

Field programmable gate arrays (FPGA) have evolved from being a flexible logic design platform to a signal-processing engine [13]. An increasing number of signal processing functions in FPGA and several capabilities like embedded memory and advanced routing. The availability of high density and high performance make them highly designable for developing hardware prototypes of communication systems.

Parameterization [14] is a new field of study derived from software radio domain. This field proposes a new approach in which similarities and differences between systems and standards must be identified and then parameterized. We can distinguish two types of parameterization either by common functions or by common operators. Parameterization by common operators, it consists to find a common operator of the highest level, which would be used by the maximum functions of many standards including future standard. Parameterization by common functions, it consists to identify the common functions of

all standards that will be implemented in reconfigurable systems.

In addition, since past research has proposed several efficient algorithms [15, 16, 17] and architectures [18,19, 20] for the RS coder-decoder that could be used as parameters for an advanced reconfigurable architectures.

This article is structured in eight sections. Section 2 provide a brief description of Reed-Solomon codes. Section 3 describes RS codec architectures and is sketching briefly the principle functionality of RS decoder. In section 4, several implementation RS coder-decoder architectures are studied. Section 5 discuss parameterization approach used in order to implement a reconfigurable RS coder-decoder for DVB, mobile and wireless systems. Section 6 explains our conception and an optimized FPGA implementation of a reconfigurable FEC systems based on RS codes used in new generation systems. Design decisions and simulation results of the verification process are reported and discussed in section 7. Finally, we summarize and conclude this article in section 8; also we propose some recommendations for future research.

2 Reed-Solomon CODECs:

Reed Solomon (RS) codes are a subset of Bose Chaudhuri-Hochquenghem (BCH) codes [21 22] and are linear block codes[23]. They are powerful error-correcting codes whose symbols are chosen from a finite field, GF(m). Their non-binary nature makes them particularly suitable to correct error bursts.

A Reed-Solomon code is specified as RS(n,k) with m-bit symbols. This means that the encoder takes k data symbols of s bits each and adds parity symbols to make an n symbol codeword. There are n-k parity symbols of s bits each. A Reed-Solomon decoder can correct up to t symbols that contain errors in a codeword, where 2t = n-k.

If the location of the symbol errors is marked as an erasure, the RS decoder can correct twice as many errors. External circuitry identifies which symbols have errors and passes this information to the decoder using the eras_sym signal. The eras_sym input indicates an erasure (when the erasures-supporting decoder option is selected).

2.1 Encoding of Reed-Solomon codes

Let (u_0, u_1, u_2,…, u_{k-1}) denote k m-bit data symbols. These symbols are encoded into a codeword (c_0, c_1, c_2,…, c_{n-1}). This encoding process is best described in terms of data polynomial:

$$I(x) = u_0 + u_1x + u_2x^2 + \ldots + u_{k-1}x^{k-1} \quad (1)$$

$$C(x) = c_0 + c_1x + c_2x^2 + \ldots + c_{n-1}x^{n-1} \quad (2)$$

C(x) are polynomial multiple of G(x), which is the generator polynomial of the code, which is defined as

$$G(x) = \prod_{i=0}^{2t-1}(x - \alpha^{(m0+i)}) \quad (3)$$

where m_0 is typically 0 or 1 Since 2t consécutive power $\alpha^{m0}, \alpha^{m0+1}, \ldots\ldots \alpha^{m0+2t-1}$ of α are rootsof G(x), C(x) is amultiple of G(x), it follows that

$$C(\alpha^{(m0+i)}) = 0, 0 \le i \le 2t-1 \quad (4)$$

for all codeword polynomials C(x). In fact,an arbitary polynomial of degree less than n is a codeword polynomial if and only if it satisfies equation 4.

Asystematic encoding produces codeword polynomials that are comprised of data followed by parity check symbols (figure 1), and it is obtained as follows (Equation 5)

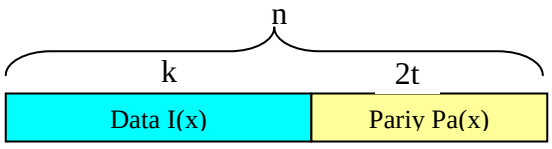

Figure 1: RS encoding

$$Pa(x) = (x^{2t} * I(x)) \bmod G(x) \quad (5)$$

It follows that the codeword is given by (C_{n-1}, $C_{n-2,}$….., $C_{1,}C_0$)= (u_{k-1}, $u_{k-2,}$….., $u_{1,}u_0$, P_{n-k-1}, $P_{n-k-2,}$….., $P_{1,}P_0$) and consists of the data symbols followed by the check symbols.

In digital hardware, the encoder is an LFSR with internal feedback connections corresponding to *G(x)*, as seen in Figure 2. The operations involved are GF addition and multiplication. The computation of the remainder is implemented on digital hardware using a linear feedback shift register configuration as shown in Figure 2. The final contents of the shift registers will contain the remainder.

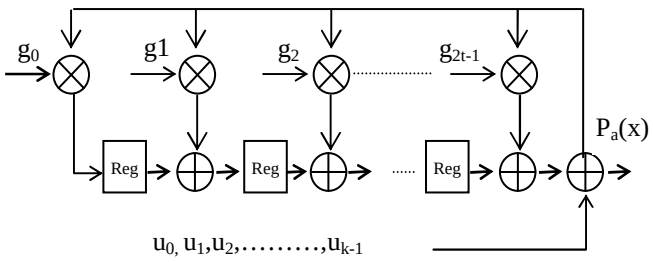

Figure 2: Typical RS encoder implementation

2.2 Decoding of Reed-Solomon codes

The general decoding steps are illustrated in Figure 3. The syndrome calculator generates a set of syndromes from the received codeword polynomial R(x). From the syndromes, the key equation solver produces the error locator polynomial σ(x) and the error evaluator polynomial Ω(*x)* which can be used by the Chien Search and the Error Value Evaluator to determine the error locations and error values, respectively.

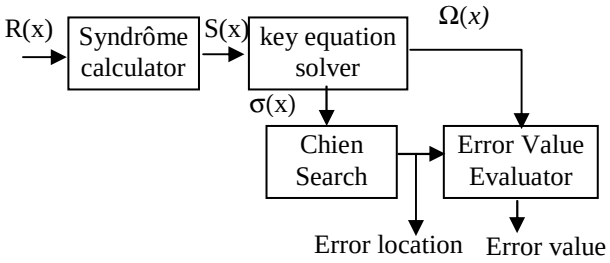

Figure 3: The simplified Reed-Solomon decoding flowchart.

3 RS decoding Algorithms study

3.1 Syndrome computation

The Syndrome calculation block treats the input codeword as a series of polynomial coefficients and calculates a syndrome polynomial of 2t coefficients. The syndrome polynomial contains the location and magnitude of up to t errors in an invalid codeword. A valid codeword generates a syndrome polynomial with all zero coefficients.

Assuming a corrupt transmission, the received codeword R consists of the original codeword which is superposed by the error E:
By definition the syndrome polynomial is S(x)

$$S(x)=\sum_{i=1}^{2t} S_i\, x^{(i-1)},\ S_i=R(\alpha^i)\ , \qquad (6)$$

$$R(x) = R_0+R_1x+ R_2x^2+\ldots..+ R_{N-1}x^{N-1} \qquad (7)$$

$$\text{then } S_i= R_0+\alpha^i(R_1+ \alpha^i(R_2+\ldots..+\alpha^i(R_{N-1}))) \quad (8)$$

where R(x) is a received polynomial and R_{N-1} is the first received symbol into a syndrome cell.
This structure describe a recursive operation multiplies and accumulates a constant value α^I with the input data. As shown in Figure 4(a), at each cycle, the partial syndrome is multiplied with and accumulated with the received symbol. After all the received symbols are processed, the accumulated result is the $i^{ième}$ syndrome. Figure 4(b) shows how the 16 syndrome cells (for t= 8) are organized in our chip. By controlling the multiplexer in Figure 4(b), we can generate different syndrome sequences for the calculation of the discrepancy in the key equation solver. Table I shows all 16 different syndrome sequences [24].

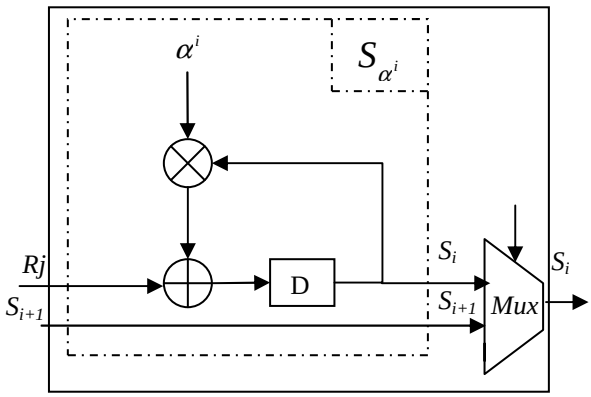

Figure 4 (a) Syndrome cell S

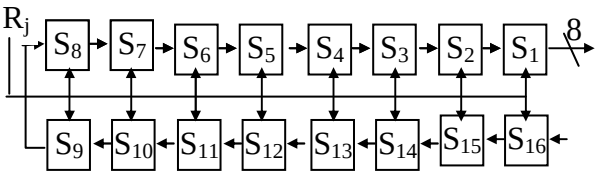

Figure 4 (a) Syndrome cell S

3.2 Key Equation Solver

The main component of an RS-decoder is the key equation calculation block. It solves a set of 2t linearly dependent equations. It generates the key equations (σ(x): locator polynomial and Ω(*x):* evaluator polynomial) from the syndrome polynomials. The locator polynomial contains information about the location of bad symbols in the codeword. The evaluator polynomial contains information about the error magnitude of the bad symbols. The two polynomials σ(x) and Ω(*x) are*

defined respectively by the following equations 9 and 10.

$$\sigma(x) = \prod_{i=1}^{w} (1 - x \cdot X_i) \quad (9)$$

where w : is the number of errors occurs in R(x).

$$\Omega(x) = \sum_{i=1}^{w} Yi Xi \prod_{j=1, j\neq i}^{w} (1 - Xj * x) \quad (10)$$

The two polynomials are related to S(x) through the Key equation (Eq11) [25, 26], so we can determine the above two unknown polynomials σ(x) and Ω(*x)* by solving the key equation:

$$S(x) * \sigma(x) = \Omega(x) \mod x^{2t} \quad (11)$$

The techniques frequently used to solve the key equation include the Berlekamp–Massey algorithm [25,27,28,29], the Euclidean algorithm [29,30]. Compared to the Euclidean algorithm, the Berlekamp–Massey algorithm is generally considered to be the one with the least hardware complexity [31].

3.2.1 Berlekamp–Massey algorithm

One of the fastest and hence often preferred algorithm is the so called "Berlekamp Massey Algorithm" (BMA) that solves

$$\sum_{w}^{2t-1} \sigma(x) S(x) = 0 , \quad (12)$$

where w ≤t is the number of errors that have occurred. So eq 12 can be developed as Eq 13

$$S_{w+1} + \sigma_1 S_w + \sigma_2 S_{w-1} + ... + \sigma_w S_1 = 0$$

$$S_{w+2} + \sigma_1 S_{w+1} + \sigma_2 S_w + ... + \sigma_w S_2 = 0$$

...

$$S_{2t} + \sigma_1 S_{2t-1} + \sigma_2 S_{2t-2} + ... + \sigma_w S_{2t-w} = 0$$

The problem of finding the minimum-degree solution to the key equation is the same as trying to find the smallest (LFSR) σ(□x)□, that generates the first 2t terms of S (figure 5).

The algorithm aims to find an LFSR of minimal length such that the first *(2t)* elements in the LFSR output sequence are the *(2t)* syndromes. The taps of this shift register are the coefficients of the desired error locator polynomial, *σ(x)* [33].

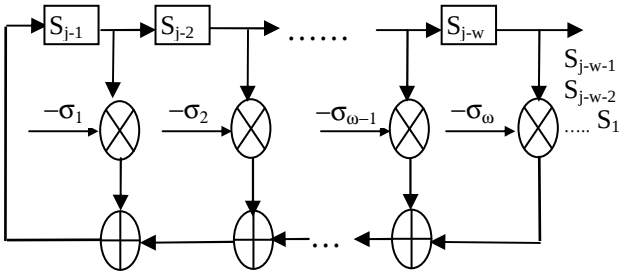

Figure 5 : Berlekamp–Massey algorithm based on LFSR implementation

$$S_j = -\sum_{i=1}^{w} \sigma_i * S_{j-i}$$

If syndrome values are known, we can compute σ(x) polynomial by the following diagram (figure6)

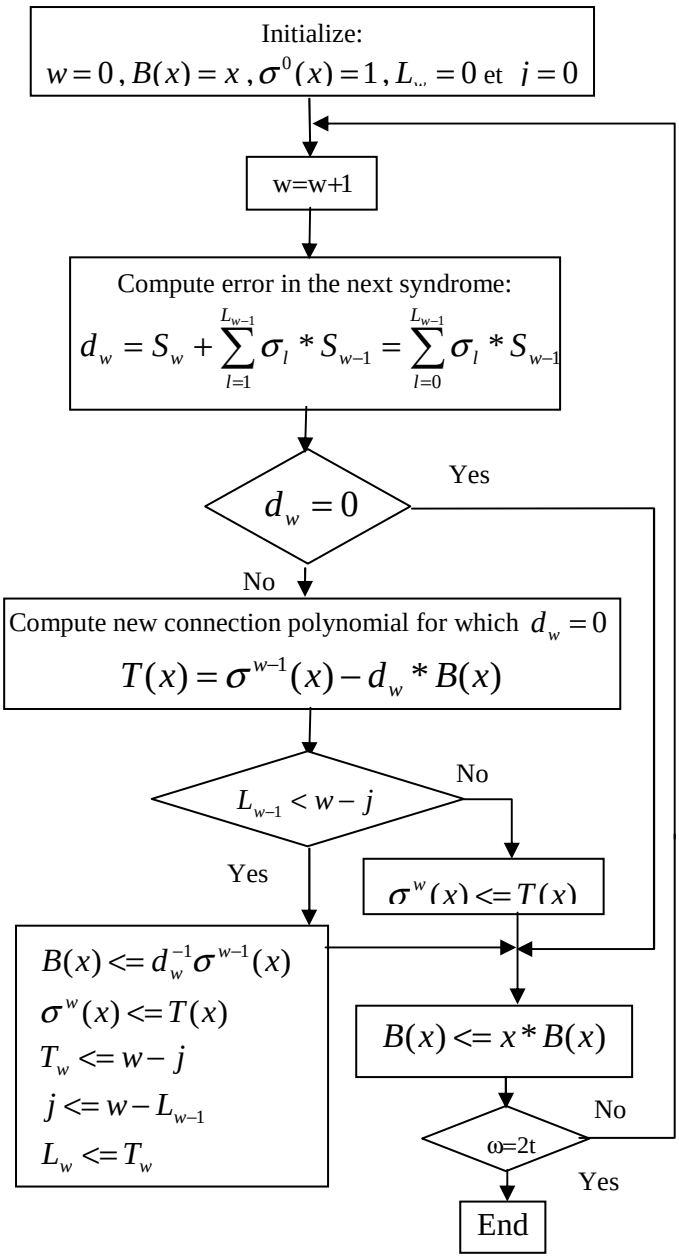

Figure 6: Barlekamp massey algorithm [32]

The implementation was a purely functional VHDL description.

3.2.2 Error Magnitudes polynomial Computing

Solving the *key equation (Eq. 11)* determines the error evaluator or error magnitude polynomial, $\Omega(x)$. An efficient way of computing $\Omega(x)$ is to perform parallel computation of $\sigma(x)$. Using the Berlekamp–Massey algorithm, this involves an iterative algorithm to compute. However, if is first obtained, from the key equation and the Newton's identity we could derive as follows:

$$\Omega(x) = S(x)\sigma(x) \bmod x^{2t}$$
$$= (S_1 + S_2x + \ldots + S_{2t}x^{2t-1}) \bullet (\sigma_1 + \sigma_2x + \ldots + \sigma_tx^t) \bmod x^{2t}$$
$$\equiv \Omega^{(0)} + \Omega^{(1)}x + \ldots + \Omega^{(t-1)}x^{t-1}$$
$$\Omega^{(i)} = S_{i+1}\sigma_0 + S_i\sigma_1 + \ldots + S_1\sigma_i, \quad i = 0, 1, \ldots t-1$$

The penalty of this efficient computation is the additional latency because $\sigma(x)$ and $\Omega(x)$ are computed in sequence.

3.3Chien Search Algorithm

With the known error locator polynomial it is possible to determine the error locations by checking whether the error locator polynomial equals zero or not. The roots of the error-locator polynomial are the inverse error locations of the codeword. To find the roots of the polynomial, a Chien Search (CS) was conducted. It uses all possible input values and then checks to see if the outputs are zero. This happens only when an error occurs. For each element that is substituted into the polynomial that equates to zero, the element is stored into memory, as these elements are the roots of the polynomial and hence, the inverse error locations.

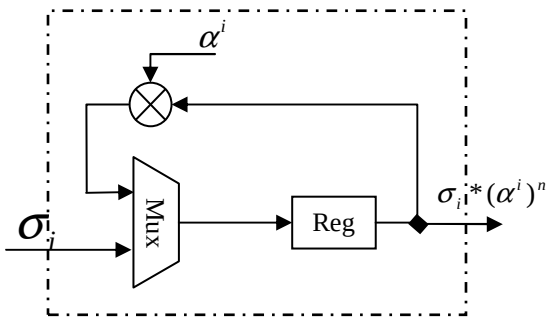

Figure 7.a Chien search cell

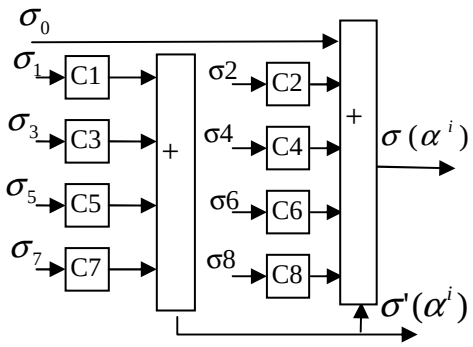

Figure 7.b Chien search structure for t = 8.

There are $(t+1)$ stages of the CS that are implemented in hardware. Each of these stages (where a stage consists of a multiplier, mux and register) (figure 7.a) represents a different value for j in the above CS equation. The search is run for n clock cycles (each clock cycle represents a different value of i in the above equation) and the output of the adder is examined to see if it is equal to zero. If it is equal to zero, the Zero Detect block will output a 1, otherwise, it will output a zero. The output of the Chien Search block is thus a string of n bits that have values of either 0 or 1. Each 1 represents the location of a symbol in error. For the first clock cycle, the mux will route the error locator polynomial coefficient into the register. For the remaining $(n - 1)$ clock cycles, the output of the multiplier will be routed via the mux into the register.

3.4Forney Algorithm

The Forney algorithm is used to compute the error values Y_i. To compute these values, the Forney algorithm needs the error locator polynomial $\sigma(x)$ *and error* magnitude polynomial $\Omega(x)$. The equation for the error values is given by Eq 14 :

$$Yi = ei = \frac{\Omega(X_i^{-1})}{\sigma'(X_i^{-1})} \text{ for } i=1\ldots t, \qquad (14)$$

where X_i^{-1} indicates roots indicates the root as computed from the Chien Search, and $\sigma'(x)$ the derivative of the error locator polynomial. Because of the fact that any element will be zero while multiplying an even constant value, and will be its original value while multiplying an odd constant, the first derivative of can be represented by :

$$\sigma'(X_i^{-1}) = \frac{1}{X_i^{-1}} \sigma odd(X_i^{-1}) \quad (15)$$

Then we can rewrite Eq 14 as the following format:

$$Yi = ei = \frac{\Omega(X_i^{-1}) X_i^{-1}}{\sigma odd(X_i^{-1})} \quad (16)$$

The $x\Omega(x)$ polynomial is then evaluated along using the same type of hardware as used for the CS. However, in order to form $x\Omega(x)$, the coefficients of $\Omega(x)$ are shifted to the left by one location.

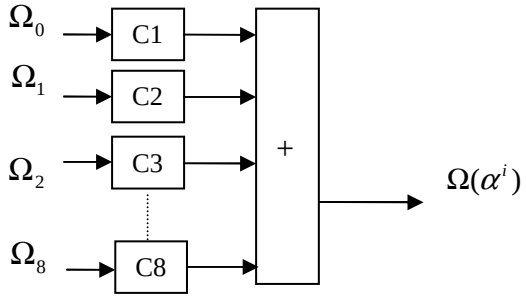

Figure 8 $\Omega(\alpha^i)$ calculator block for t = 8.

The numerator is then multiplied by the denominator using an inverse multiply. The inverse multiply contains a lookup table that finds the inverse of the denominator. For example, if the denominator was α^3, the inverse is α^{-3}. This can then be expressed as: $\alpha^{-i} = \alpha^{(-i \bmod n)} = \alpha^{(-3 \bmod 255)} = \alpha^{252}$.

Since the same type of hardware is needed for both the Chien Search and the Forney algorithm. The output of the adder for the odd stages is also used in the Forney algorithm, shown in the middle part of the figure 9. The sum of the odd stages represents the denominator of the Forney equation. This value is inversed in the Inverse Multiply block and then multiplied by the numerator value that is formed from evaluating the error magnitude polynomial. The output is "ANDed" with the zero detect output since the error values are only valid for the actual error locations (and they should be set to zero otherwise).

Once the error magnitudes are calculated, the error corrector block takes the received code and performs XOR-operation with the corresponding error magnitudes computed at the respective error locations to attain the original message stream (Eq. 17).

$$C(X_i) = R(X_i) \oplus Y_i \quad (17)$$

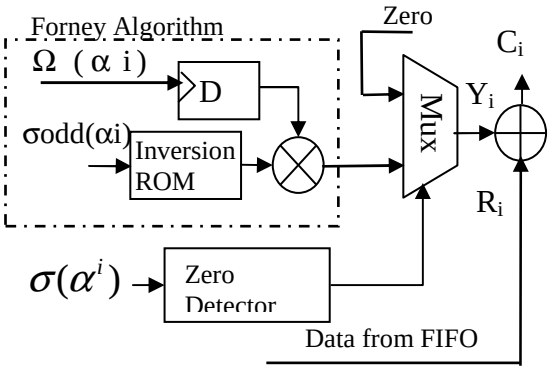

Figure 9 Error value evaluator structure for t = 8.

4 RS parameterization approach for new generation system

Past research has proposed several efficient RS algorithms and sub architectures. On the other hand there are others research works, which are interested on compiler development [34, 35]. However in the literature there are a few researches that are focused on implementing reconfigurable RS coder-decoder. K.SHIMIZU & N. TOGAWA have proposed a reconfigurable adaptive FEC System based on RS code with interleaving. In adaptive Scheme error correction capability t is changed dynamically according to the communication channel conditions. The packet error rate is employed as threshold value to change t [36].

In this section we define an advanced RS coder-decoder architecture based on parameterization approach which a key solution for software defined radio (SDR) systems. Our parameterization approach is used in order to implement on FPGA a generic RS coder-decoder for DVB and wireless systems.

Generic RS module must integrate all common RS configuration, which are used in most mobile and wireless systems. Implementing a generic RS module, match up parameterisation by common functions approach. Different parameters can have generic value in configurable RS module. Fully parameterized RS function, including:

- Number of bits per symbol
- Number of symbols per codeword
- Number of check symbols per codeword

- Field polynomial
- First root of generator polynomial

The symbol width (m) defines the field generator polynomial. Table 1 illustrates this correspondence.

Symbol Width	Field generator polynomial
3	x^3+x+1
4	x^4+x+1
5	x^5+x^2+1
6	x^6+x+1
7	x^7+x^3+1
8	$x^8+x^4+x^3+x^2+1$
9	x^9+x^4+1
10	$x^{10}+x^3+1$
11	$x^{11}+x^2+1$
12	$x^{12}+x^6+x^4+x+1$

Table 1: Correspondence between symbol width and the field generator polynomial

All most new generation standards use eight value as symbol width this leads to use $x^8+x^4+x^3+x^2+1$ as a field generator polynomial.

4.1 RS parameterization for DVB norm

DVB has three standards that use identical RS code parameters. These are satellite (DVB-S), cable (DVB-C) and terrestrial (DVB-T). The most widely used of the three protocols is DVB-S [Sohi20001.

All DVB standards employ the same (204,188) RS code. All DVB standards operate in GF(2^8), and are based on a (255,239) RS code. Therefore, the same Galois Field arithmetic units and hardware can be used for different DVB standards. Table 2 summarize RS parameters for DVB.

Parameter	Symbol	DVB
Field Polynomial	P(x)	$X^8+X^4+X^3+X^2+1$
Generator polynomial	G(x)	$(x-\alpha^0)(x-\alpha^1)(x-\alpha^2)\ldots(x-\alpha^{15})$
Bits number/Symbol	m	8
Code length	n	204
Message length	k	188
Parity Symbols	2t	16

Table 2: RS parameters for DVB

4.2 RS parameterization for Wireless 802.16

IEEE Std. 802.16 specifies the outer code requirements for RS code as follows:

The specified code generator polynomials are given by:

- *Code Generator Polynomial:*

$g(x) = (x+\mu^1)(x+\mu^2) \ldots (x+\mu^{2T})$, where μ=02hex

- *Field Generator Polynomial:*

$p(x) = x^8 + x^4 + x^3 + x^2 + 1$

The specified code has a block length of 255 bytes and shall be configured as an RS(255,255-R) code with information bytes preceded by (255-*K*) zero symbols, where *K* is the codeword length and *R* the number of redundancy bytes ($R = 2*T$ ranges from 2 to 32, inclusive). The value of *K* and *T* are specified for each burst profile by the MAC. [37]

The variable decoder supports real-time changing of the number of symbols in the codeword, and *R*, the number of check symbols in a codeword.

5 FPGA implementation of a configurable FEC systems based on RS codes for DVB and 802.16 network

Overall hardware implementation of the controller design consists of the entry of the conceptual design into electronic description format (design entry), conversion of the design into a logic level form (synthesis), and translation of the design into the physical FPGA specific component placement and signal routing (implementation). The design verification process consists of testing the design for conformity at several intermediate stages. The verification steps performed after each major stage of the design are shown in figure 10 and include: behavioral or functional simulations, synthesis checks, postsynthesis timing verification, and post-implementation timing verification. All of these steps are done using simulation tools like Xilinx's Foundation ISE Tools [38] and Modelsim XE 6.0d.