Conception of a new Syndrome Block for BCH codes with hardware Implementation on FPGA Card

El Habti El Idrissi Anas\textsuperscript{1}, El Gouri Rachid\textsuperscript{1, 2}, Ahmed Lichioui\textsuperscript{3}, Hlou Laamari\textsuperscript{1} \\
\textsuperscript{1}Laboratory of Electrical Engineering and Energy Systems, Faculty of Sciences, University Ibn Tofail Kenitra, Morocco \\
\textsuperscript{2}National School of Applied Sciences, University Ibn Tofail Kenitra, Morocco \\
\textsuperscript{3}National Society of Radio and Television (SNRT), Rabat, Morocco

ABSTRACT
Error Correcting Codes are required to have a reliable communication through a medium that has an unacceptable bit error rate and low signal to noise ratio. Data gets corrupted during the transmission and reception due to noises and interferences. The Bose, Chaudhuri, and Hocquenghem (BCH) codes are being widely used in variety communication and storage systems. In this paper, a simplified algorithm for BCH decoding is proposed with a view to reduce the number of iterations for error detection in the syndrome calculator block of BCH decoders with a percentage of up to 80\% compared to the basic syndrome block. First, we developed the design of the proposed algorithm second, we generated and simulated the hardware description language source code using Quartus software tools and finally we implemented the new algorithm of syndrome block on FPGA card.

Keywords: Digital video broadcasting-satellite-second generation, BCH decoder, Syndrome Block, iterations, add syndromes, FPGA implementation.

I. INTRODUCTION
Error correcting codes are used in satellite communication, cellular telephone networks, body area networks and in most of the digital applications. There are different types of error correcting codes based on the type of error expected, expected error rate of the communication medium, and whether retransmission is possible or not. Few of them are BCH, Turbo, Reed Solomon, Hamming and LDPC. These codes differ from each other in their implementation and complexity [1]. BCH codes are large class of multiple error-correcting codes. The binary BCH codes were discovered in 1959 by Hocquenghem and independently in 1960 by Bose and Ray-Chaudhuri. Later, Gorenstein and Zierler generalized them to all finite fields [2].

In practice, the binary BCH and Reed-Solomon codes are the most commonly used variants of BCH codes. They have applications in a variety of communication systems: digital subscriber loops, wireless systems, networking communications, and magnetic and data storage systems. Continuous demand for ever higher data rates and storage capacity provide an incentive to devise very high-speed and space-efficient VLSI implementations of BCH decoders. The first decoding algorithm for binary BCH codes was devised by Peterson in 1960. The objective of this work is to reduce a large number of iterations in the syndrome Block using a new algorithm which allows conceiving another circuit of syndrome block. The rest of the paper is organized as follows: an overview of syndrome calculator block of BCH decoders is Provided in section 2. Section 3 discusses the proposed algorithm and simulation. Finally, FPGA implementation details, hardware performance results are presented in section 4, followed by a conclusion.

II. BACKGROUND AND RELATED WORK
The BCH code is characterized as (n, k, t), where n is the code length, k is the data length, and t is the error correction capability. The n-bit code word $r_0, r_1, ..., r_{n-1}$ can be interpreted as a received polynomial [3]. $R(x) = r_0 + r_1 x + r_2 x^2 + ... + r_{n-1} x^{(n-1)}$. In Syndrome Calculation, 2t syndromes are computed using the following equation:

$$S_i = R(a^i) = \sum_{j=0}^{n-1} r_j a^{ij} = r_0 + r_1 a^i + r_2 a^{2i} + ... + r_{n-1} a^{(n-1)i} \quad (1)$$

The syndrome calculator block provides two results. The first result is whether the received code word is correct. The second result provides the syndrome polynomial which will be used to correct the code word if the code word is erroneous. The most common algorithm to perform the syndrome calculation needs 2t basic cells as defined in Fig.1. Where 1≤i≤2t. For each Syndrome $S_i$, where 1≤i≤2t, n iterations or computations are needed to compute the Syndrome Polynomial. All the syndrome coefficients will be equal to 0 if the received code word is correct with at least one coefficient different from 0 if the code word is not correct. Much work has been done to reduce time-consuming computational
steps. Costa et al. [4] developed a new fast Fourier transform (FFT) method based on the cyclotomic FFT to compute the syndrome using a method that corresponds to the partial discrete Fourier transform (DFT) of \(D(x)\). This algorithm is better than the Horner rule and the Zakharova method.

\[
R(x) = r_{254}x^{254} + r_{253}x^{253} + \ldots + r_2x^2 + r_1x + r_0x^0 (5)
\]

\[
S_j(j=5) = \sum_{i=0}^{n-5} stepof_j A_i \cdot \alpha^i = A_0 + A_1\alpha^5 + A_2\alpha^{10} + \ldots + A_5\alpha^{25} (6)
\]

Where:

\[
A_0 = r_0 + r_51 + r_102 + r_153 + r_254
A_1 = r_1 + r_52 + r_103 + r_154 + r_205
A_2 = r_2 + r_53 + r_106 + r_155 + r_206
A_3 = r_3 + r_107 + r_152 + r_203 + r_254
\]

So we were able to reduce the 356 iterations (basic circuit) to 52 iterations (modified circuit) despite of the possibility of having more logic gates for the new method. For equation (2) the basic circuit corresponding is represented in Fig.1. The simulation of the basic circuit (equation 2) using Quartus is represented in Fig.2. For the equation (4) the modified circuit using the factorization method is represented in Fig.3. The simulation of the modified circuit (equation 4) using Quartus is represented in Fig.4.

3.1) Syndrome block used in DVB-S2

The DVB-S2 standard lists includes 21 variants of long BCH codes. Each variant is identified by its code block size \(N_{\text{ch}}\), uncoded block size \(K_{\text{ch}}\), error correction capability \(t\) and frame type. Table 1 represents some examples of the 21 variants listed in tables 5a and 5b of the specifications [8].

<table>
<thead>
<tr>
<th>(K_{\text{ch}})</th>
<th>(N_{\text{ch}})</th>
<th>(t)</th>
<th>Frame</th>
</tr>
</thead>
<tbody>
<tr>
<td>16008</td>
<td>16200</td>
<td>12</td>
<td>Normal</td>
</tr>
<tr>
<td>51648</td>
<td>51840</td>
<td>12</td>
<td>Normal</td>
</tr>
<tr>
<td>53840</td>
<td>54000</td>
<td>10</td>
<td>Normal</td>
</tr>
<tr>
<td>58192</td>
<td>58320</td>
<td>8</td>
<td>Normal</td>
</tr>
<tr>
<td>3072</td>
<td>3240</td>
<td>12</td>
<td>short</td>
</tr>
</tbody>
</table>

The codes for normal frames are computed in GF \((2^8)\), whereas the short frame codes are computed over GF \((2^{14})\). The primitive polynomials used to generate the Galois fields are:

\[
x^{16} + x^4 + x^3 + x + 1 \text{ for } GF (2^8) \\
x^{14} + x^3 + x^2 + x + 1 \text{ for } GF (2^{14})
\]

For the case of BCH (3240, 3072, 12) where:

\(N=3240, k=3072\) and \(t=12\) we have \(2t = 24\) Syndromes: \(S_1, S_2, S_3, S_4, S_5, \ldots, S_{24}\). \(S_1\) can be calculated by the direct method and the even syndromes can be calculated using the following relationship \(S_{2i} = S_i^2\). For example, to calculate the odd syndrome \(S_1\) we use the proposed algorithm as follows:

\[
S_j(j=9) = \sum_{i=0}^{n-1j} stepof_j A_i \cdot \alpha^i = A_0 + A_1\alpha^9 + A_2\alpha^{18} + \ldots + A_{359}\alpha^{3231}
\]

![Image](https://via.placeholder.com/150)

**Figure 1. Basic syndrome calculator cell**

### III. PROPOSED ALGORITHM

The proposed algorithm is based on a change [5, 6] of the receiver code word such as \((j divided n, j is an odd number)\) in the code BCH \((n, k, 1)\) to calculate the add syndromes \(S_j\). Besides, with this method we can conceive another circuit of the Syndrome Block i.e. we can reduce latency by decreasing the number of iterations compared to the basic circuit. For instance, in the case of BCH \((15, 7, 2)\) where: \(N=15, k=7\) and \(t=2\) we have \(2t = 4\) Syndromes: \(S_1, S_2, S_3, S_4\).

\[
R(x) = r_{14}x^{14} + r_{13}x^{13} + \ldots + r_1x^1 + r_0
\]

The received code word. Knowing that \(S_j\) can be calculated by the direct method and the even syndromes can be calculated using the following relationship \(S_{2i} = S_i^2\). To calculate the odd syndromes [7] we use the proposed algorithm as follows:

\[
S_j = R(\alpha^j) = R_1\alpha^{j2} + R_2\alpha^{j3} + R_1\alpha^{j4} + R_2\alpha^{j5} + R_1\alpha^{j6} + R_2\alpha^{j7} + R_1\alpha^{j8} + R_2\alpha^{j9} + R_1\alpha^{j10} + R_2\alpha^{j11} + R_1\alpha^{j12} + R_2\alpha^{j13} + R_1\alpha^{j14} + R_2\alpha^{j15} + R_1\alpha^{j16} + R_2\alpha^{j17} + R_1\alpha^{j18} + R_2\alpha^{j19} + R_1\alpha^{j20} + R_2\alpha^{j21} + R_1\alpha^{j22} + R_2\alpha^{j23} + R_1\alpha^{j24} + R_2\alpha^{j25} + R_1\alpha^{j26} + R_2\alpha^{j27} + R_1\alpha^{j28} + R_2\alpha^{j29} + R_1\alpha^{j30} + R_2\alpha^{j31} + R_1\alpha^{j32} + R_2\alpha^{j33} + R_1\alpha^{j34} + R_2\alpha^{j35} + R_1\alpha^{j36} + R_2\alpha^{j37} + R_1\alpha^{j38} + R_2\alpha^{j39} + R_1\alpha^{j40} + R_2\alpha^{j41} (3)
\]

Where:

\[
A_i = R_1+R_2+R_1R_2
B = R_1+R_2+R_1R_2
C = R_1+R_2+R_1R_2
D = R_1+R_2+R_1R_2
E = R_1+R_2+R_1R_2
\]

So we were able to reduce the sixteen iterations (basic circuit) to six iterations (modified circuit) despite of the fact that we may have more logic gates for the new method. For the case of BCH \((255, 231, 3)\) where: \(N=255, k=231\) and \(t=3\) we have \(2t = 6\) Syndromes: \(S_1, S_2, S_3, S_4, S_5, S_6\). \(S_{2i}\) can be calculated by the direct method and the even syndromes can be calculated using the following relationship \(S_{2i} = S_i^2\). For example, to calculate the odd syndrome \(S_3\) we use the proposed algorithm as follows:

\[
S_j(j=9) = \sum_{i=0}^{n-1j} stepof_j A_i \cdot \alpha^i = A_0 + A_1\alpha^9 + A_2\alpha^{18} + \ldots + A_{359}\alpha^{3231}
\]
\[ A_0 = r_0 + r_{360} + r_{720} + r_{1080} + r_{1440} + r_{2160} + r_{2520} + r_{2880} \]
\[ A_1 = r_1 + r_{361} + r_{721} + r_{1081} + r_{1441} + r_{2161} + r_{2521} + r_{2881} \]
\[ A_2 = r_2 + r_{362} + r_{722} + r_{1082} + r_{1442} + r_{2162} + r_{2522} + r_{2882} \]
\[ A_{358} = r_{358} + r_{718} + r_{1078} + r_{1438} + r_{2158} + r_{2518} + r_{2878} + r_{3238} \]
\[ A_{359} = r_{359} + r_{719} + r_{1079} + r_{1439} + r_{2159} + r_{2519} + r_{2879} + r_{3239} \]

So we were able to gain 2880 iterations, 3241 iterations (basic circuit) to 361 iterations (new method) despite of the fact that we may have more logic gates for the new method. The same as \( S_j (j=3), S_j (j=5) \) and \( S_j (j=15) \). Generally for a code characterized as \((n, k, t)\), where \(n\) is the code length, \(k\) is the data length, \(t\) is the error correction capability and \(j\) divides \(n\)

We have:

\[
S_j = \sum_{i=0, \text{stepof} j}^{n-j} A_i \times \alpha^i
\]
\[
A_i = \sum_{i=1, \text{stepof} \ (n/i)}^{n} r_i
\]

Where:

- \( j \) is an odd number
- \( j \) divides \( n \)
- \( S_j \) are the odd syndromes

The corresponding logic circuit is represented in Fig.5.
3.2) **Comparison of circuits**

According to the simulation we have adopted previously, the result we got is the same one in comparison with the modified circuit but with an important number of iterations. This reduction of iterations can reach 80% compared to the basic circuit. The table 2 shows the number of the gained iterations by using both the basic circuit and the modified circuit for different BCH codes.

<table>
<thead>
<tr>
<th>BCH Codes</th>
<th>Number of syndromes</th>
<th>value of J</th>
<th>Number of iterations for basic circuit</th>
<th>Number of iterations for modified circuit</th>
<th>Number of gained iterations</th>
</tr>
</thead>
<tbody>
<tr>
<td>BCH (15, 7, 2)</td>
<td>2t = 4</td>
<td>J = 3</td>
<td>16</td>
<td>6</td>
<td>10</td>
</tr>
<tr>
<td>BCH (255, 231, 3)</td>
<td>2t = 6</td>
<td>J = 3</td>
<td>256</td>
<td>86</td>
<td>170</td>
</tr>
<tr>
<td></td>
<td></td>
<td>J = 5</td>
<td>256</td>
<td>52</td>
<td>204</td>
</tr>
<tr>
<td></td>
<td></td>
<td>J = 9</td>
<td>256</td>
<td>361</td>
<td>2880</td>
</tr>
<tr>
<td></td>
<td></td>
<td>J = 15</td>
<td>256</td>
<td>217</td>
<td>3024</td>
</tr>
<tr>
<td>BCH (n, k, t)</td>
<td>2t</td>
<td>J</td>
<td>n+1</td>
<td>(n/J) + 1</td>
<td>n. ((J-1)/J)</td>
</tr>
</tbody>
</table>
IV. FPGA IMPLEMENTATION

Implementation of decoders for BCH codes has been problematic due to the very large number of resources required. Our quest is to implement a new Syndrome Block on FPGA based on the proposed algorithm to judge the savings in hardware resources [6, 7]. In this paper a parameterized hardware model of the Syndrome Block was developed using the Hardware Description Language (VHDL) and synthesized using Xilinx Synthesis Tool. The block diagram of the proposed algorithm as implemented is shown in Fig.6. The proposed Syndrome Block consists of a global ‘Clk’ and the error detection process is initiated by an ‘Input’, the ‘Syndromes Sj’ can be obtained immediately after entering input.

![Figure 6: Block diagram of Syndrome Block](image)

Figure 6. Block diagram of Syndrome Block

V. COMPARISON OF ALGORITHMS

According to the tables 3, 4 and 5 which indicate the FPGA device utilization summary for BCH (15, 7) and BCH (255, 231) codes by using the two algorithms, we can conclude that: The proposed algorithm presents a low complexity and a very good performance compared to the basic algorithm.

Table 3. FPGA device utilization summary for BCH (15, 7, 2)

<table>
<thead>
<tr>
<th>LogicUtilisation</th>
<th>Device Utilisation Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of Slices</td>
<td>Used</td>
</tr>
<tr>
<td>Number of Slices Flip Flpos</td>
<td>38</td>
</tr>
<tr>
<td>Number of input LUTs</td>
<td>46</td>
</tr>
<tr>
<td>Number of bondedIOBs</td>
<td>9</td>
</tr>
<tr>
<td>Number of GCLKs</td>
<td>1</td>
</tr>
</tbody>
</table>

Table 4. Timing summary for BCH (15, 7, 2)

<table>
<thead>
<tr>
<th>Timing Summary</th>
<th>Basic circuit</th>
<th>Modified circuit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Minimum input arrival time before clock (ns)</td>
<td>2.992</td>
<td>2.637</td>
</tr>
<tr>
<td>Maximum output required time after clock (ns)</td>
<td>7.233</td>
<td>6.580</td>
</tr>
<tr>
<td>Maximum combinational path delay (ns)</td>
<td>8.012</td>
<td>7.302</td>
</tr>
</tbody>
</table>

Table 5. FPGA device utilization summary for BCH (255, 231, 3)

a) FPGA device utilization summary for Basic circuit

<table>
<thead>
<tr>
<th>Logic Utilisation</th>
<th>Device Utilisation Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of Slices Registers</td>
<td>Used</td>
</tr>
<tr>
<td>Total Number of Slices</td>
<td>56</td>
</tr>
<tr>
<td>Number used as Flip Flpos</td>
<td>50</td>
</tr>
<tr>
<td>Number used as Latches</td>
<td>6</td>
</tr>
<tr>
<td>Number of 4 input LUTs</td>
<td>114</td>
</tr>
<tr>
<td>Number of occupied Slices</td>
<td>74</td>
</tr>
<tr>
<td>Number of Slices containing only related logic</td>
<td>74</td>
</tr>
<tr>
<td>Number of Slices containing unrelated logic</td>
<td>0</td>
</tr>
<tr>
<td>Total Number of 4 input LUTs</td>
<td>144</td>
</tr>
<tr>
<td>Number used as a route-thru</td>
<td>144</td>
</tr>
<tr>
<td>Number of bondedIOBs</td>
<td>13</td>
</tr>
<tr>
<td>Average Fanout of Non-Clock Nets</td>
<td>3.15</td>
</tr>
</tbody>
</table>
VI. CONCLUSION

In this paper, we have presented a simplified algorithm of Syndrome Block for BCH codes. This algorithm is based on a simple change of the received code word in order to reduce the number of iterations, the comparison drawn between circuits in Table 1 shows that the BCH code (J = 15) used in DVB S2 has 217 iterations using the modified method instead of 3241 iterations using the basic method. In Table 4 the timing summary for BCH (15, 7, 2) shows that the decrease of the number of iterations reduces latency 8.012ns to 7.302ns and presents a slight increase in hardware resources compared to the basic algorithm. The proposed algorithm has been implemented on a Xilinx Spartan 3E-500 FG 320 FPGA (xc3s500e-5fg320).

ACKNOWLEDGEMENTS

This work was supported by the Laboratory of Electrical Engineering and Energy System, Faculty of Science, University Ibn Tofail Kenitra, the National Society of Radio and Television (SNRT), Rabat and the National School of Applied Sciences, University Ibn Tofail Kenitra, Morocco.

REFERENCES

[8] ETSI EN 302 307, Section 5.3 FEC encoding.