# A Synthesizable BCH Decoder For DVB-S2 Satellite Communications

Cesar G. Chaves, Eduardo R. de Lima and Jacqueline G. Mertes

Department of Hardware Design, Eldorado Research Institute, Campinas, Brasil
e-mail: cesar.arroyave@eldorado.org.br

**ABSTRACT**

This paper presents the design of a BCH Decoder for digital satellite TV Communications. It includes an architecture design specification, as well as the results of FPGA prototyping and of the logical and physical synthesis in 65nm CMOS. Moreover, it can be used as a basis for BCH Decoder designs for other kind of communications or even storage error correction.

**Index Terms:** BCH, Error-correction, DVB-S2, VLSI, FPGA

## I. INTRODUCTION

The Second Generation Digital Video Broadcasting System for Satellite broadcasting and unicasting (DVB-S2) was specified to cope with any existing satellite transponder characteristics. It is used for applications such as: broadcast for High Definition Television (HDTV), iterative services for consumer applications, Digital TV distribution and news gathering, distribution of signal to terrestrial transmitters, among others [1].

The DVB-S2 standard has been specified around three key concepts: best transmission performance, total flexibility and reasonable receiver complexity that allows different modulations (QPSK, 8-PSK, 16-QAM and 32-QAM) and error protection levels (1/4, 1/3, 2/5, 1/2, 3/5, 2/3, 3/4, 4/5, 5/6, 8/9, and 9/10) to be used on a frame by frame basis, achieving about 30% of capacity gain over its predecessor [1].

Due to an improved Forward Error Correction (FEC) subsystem, the DVB-S2 standard achieves a near Shanon limit [2] performance. This subsystem is composed by the De-Interleaver, Low Density Parity Check Code (LDPC) decoder [3], [4] and the Bose-Chaudhuri-Hocquenghem (BCH) decoder [5]–[8] blocks.

The De-interleaver rearranges the data in the original sequence, previously interleaved in a non-contiguous way to increase performance in error-correction coding. The structure of the frame provided by the De-Interleaver with its components is illustrated in Fig. 1. The length of each frame segment depends on the adopted code-rate and frame type, as established in [1].

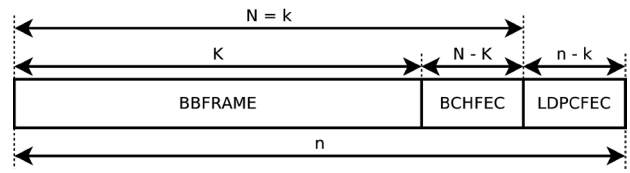


**Figure 1.** DVB-S2 FECFRAME

The high error-correction capability of DVB-S2 is achieved due to the concatenation of LDPC and BCH decoders. The former uses the LDPCFEC bits to execute a coarse correction, eliminating most of the errors. The latter uses the BCHFEC bits for doing a fine correction of up to 12 bits according to the code-rate and frame type.

The BCH decoder, focus of this paper, executes the error detection and correction process illustrated by the flowchart in Fig. 2.

The internal architecture of the proposed BCH decoder is explained along this paper. It also presents results regarding FPGA implementation, as well as logical and physical synthesis results obtained using a 65nm CMOS library. A comparison with the approach in [7] is also provided.

The remainder of the paper is organized as follows: Section II introduces the proposed BCH decoder architecture. Section III presents the implementation results. Section IV concludes the paper and outlines future work.

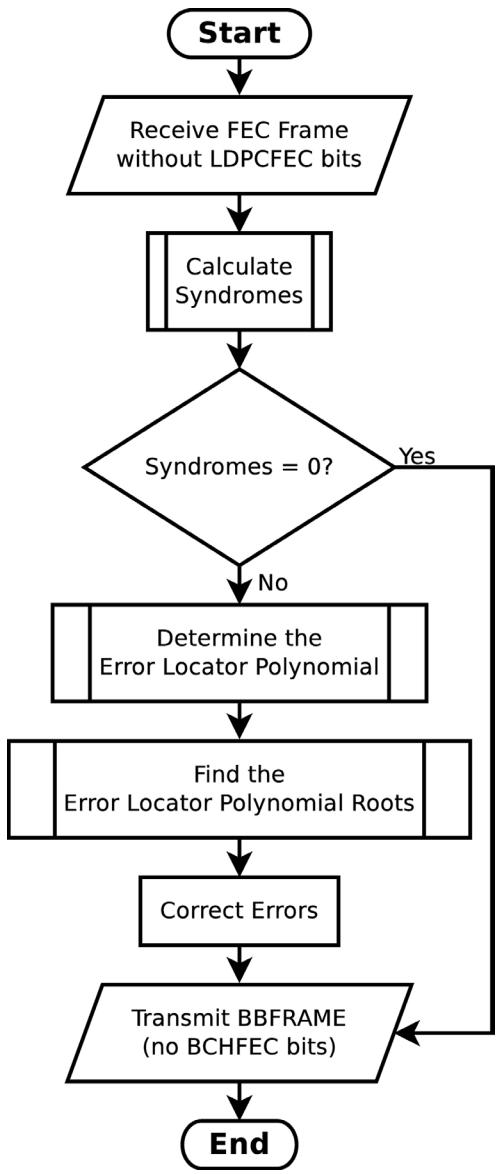


**Figure 2.** BCH Decoder Flowchart

## II. PROPOSED ARCHITECTURE

This section presents the design of a circuit able to detect and correct errors within a DVB-S2 frame. It has an error correction capacity (t) of up to 12 errors, according to the transmission code rate and frame size, as required in the DVBS2 standard [1].

The proposed BCH decoder is composed by five internal modules and an XOR gate. Its internal architecture is illustrated by Fig. 3 and the functionality of its modules explained in subsections A through E. The implementation of this design is in charge of serially receiving a FECFRAME, without LDPCFEC bits, through the data in input and transmitting an uncoded BBFRAME via the data out output.

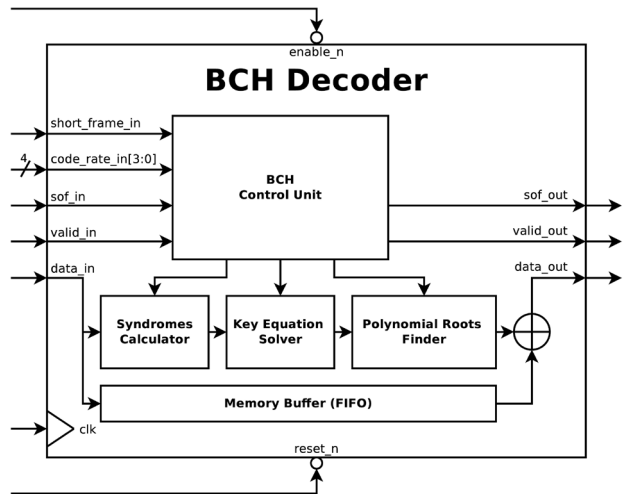


**Figure 3.** Internal architecture of the proposed BCH Decoder

### *A. The BCH Control Unit (CU)*

This module contains the counters that are set according to the frame type and code rate configurations for controlling the, also incorporated, finite state machine that coordinates the entire BCH decoding process, as shown in Fig. 4.

The BCH decoder remains in the IDLE state while no data has been received. As soon as the first valid bit arrives, the decoder switches to the RX Stage 1 state, where it receives all the bits of the BBFRAME. These bits are introduced into the Syndromes Calculator (SC) and stored in the Memory Buffer (MB) for further transmission. When all the bits of the BBFRAME have been received (i.e. first K bits), a transition to the RX Stage 2 state is done. In this state, the received bits are only inputted to the SC, this because only the first K bits need to be transmitted. When all the N bits are received, the SC will have calculated the syndromes. As illustrat-

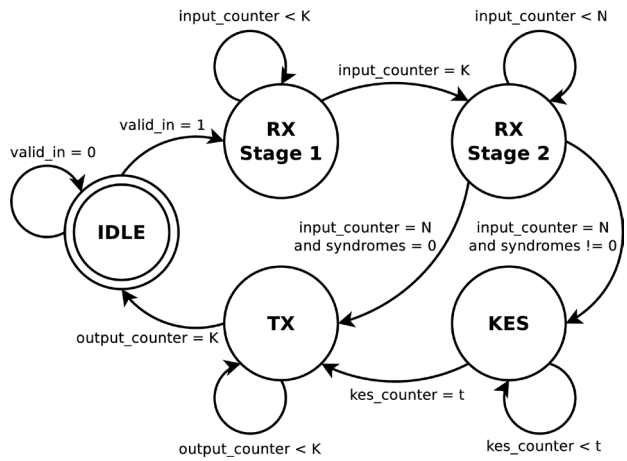


**Figure 4.** Finite State Machine of the DVB-S2 BCH Decoder

ed in Fig. 2, if all the syndromes have a zero value, no error was detected and the stored BBFRAME can be transmitted, this is done in the TX state. On the other hand, if at least, a non-zero syndrome was obtained, all 2t syndromes are passed to the key equation solver (KES) module in the KES state. After running the KES, during t clock cycles, an error locator polynomial is obtained and passed to the Polynomial Roots Finder (PRF) module. Now, while in the TX state, as bits are fetched from the MB, the PRF provides a single-bit signal when a root of the error locator polynomial is found. These two bits are XORed and the result is the output of the decoder. When all the K bits have been transmitted, the BCH decoder returns to IDLE state.

### *B. The Syndrome Calculator (SC)*

This module determines if the received frame contains or not any erroneous bits. As explained in [9], this can be done by representing the frame as a polynomial and according to its size, calculating the remainder of dividing it by the primitive polynomials from Table I or Table II, normal and short frames respectively [1].

Alternatively to the iterative design in [9], a parallel design can also be found in the literature. Nevertheless, we adopted the iterative design, which despite its larger latency, occupies less die area. Moreover, its latency can be neglected by overlapping it with the latency of data reception, so an effective latency of frame-length clock cycles is maintained instead of twice that time.

Fig. 5 depicts the architecture of the implemented SC, where the twelve Linear Feedback Shift Registers (LFSRs) are configured with the primitive polynomials from Table I or Table II, according to the size of the incoming frame. The combinational logic is composed by a set of XOR gates determined by the Algorithm 1, as explained in [9].

**Table I.** BCH polynomials for normal FEC frames.

| | |
|---|---|
| $G_1$ | $x^{16} + x^5 + x^3 + x^2 + 1$ |
| $G_2$ | $x^{16} + x^8 + x^6 + x^5 + x^4 + x + 1$ |
| $G_3$ | $x^{16} + x^{11} + x^{10} + x^9 + x^8 + x^7 + x^5 + x^4 + x^3 + x^2 + 1$ |
| $G_4$ | $x^{16} + x^{14} + x^{12} + x^{11} + x^9 + x^6 + x^4 + x^2 + 1$ |
| $G_5$ | $x^{16} + x^{12} + x^{11} + x^{10} + x^9 + x^8 + x^5 + x^3 + x^2 + x + 1$ |
| $G_6$ | $x^{16} + x^{15} + x^{14} + x^{13} + x^{12} + x^{10} + x^9 + x^8 + x^5 + x^3 + x^2 + x + 1$ |
| $G_7$ | $x^{16} + x^{15} + x^{13} + x^{11} + x^{10} + x^9 + x^8 + x^6 + x^5 + x^2 + 1$ |
| $G_8$ | $x^{16} + x^{14} + x^{13} + x^{12} + x^9 + x^8 + x^6 + x^5 + x^2 + x + 1$ |
| $G_9$ | $x^{16} + x^{11} + x^{10} + x^9 + x^7 + x^5 + 1$ |
| $G_{10}$ | $x^{16} + x^{14} + x^{13} + x^{12} + x^{10} + x^8 + x^7 + x^5 + x^2 + x + 1$ |
| $G_{11}$ | $x^{16} + x^{13} + x^{12} + x^{11} + x^9 + x^5 + x^3 + x^2 + 1$ |
| $G_{12}$ | $x^{16} + x^{12} + x^{11} + x^9 + x^7 + x^6 + x^5 + x + 1$ |

**Table II.** BCH polynomials for short FEC frames.

| | |
|---|---|
| $G_1$ | $x^{14} + x^5 + x^3 + x + 1$ |
| $G_2$ | $x^{14} + x^{11} + x^8 + x^6 + 1$ |
| $G_3$ | $x^{14} + x^{10} + x^9 + x^6 + x^2 + x + 1$ |
| $G_4$ | $x^{14} + x^{12} + x^{10} + x^8 + x^7 + x^4 + 1$ |
| $G_5$ | $x^{14} + x^{13} + x^{11} + x^9 + x^8 + x^6 + x^4 + x^2 + 1$ |
| $G_6$ | $x^{14} + x^{13} + x^9 + x^8 + x^7 + x^3 + 1$ |
| $G_7$ | $x^{14} + x^{13} + x^{11} + x^{10} + x^7 + x^6 + x^5 + x^2 + 1$ |
| $G_8$ | $x^{14} + x^{11} + x^{10} + x^9 + x^8 + x^5 + 1$ |
| $G_9$ | $x^{14} + x^{10} + x^9 + x^3 + x^2 + x + 1$ |
| $G_{10}$ | $x^{14} + x^{12} + x^{11} + x^9 + x^6 + x^3 + 1$ |
| $G_{11}$ | $x^{14} + x^{12} + x^{11} + x^4 + 1$ |
| $G_{12}$ | $x^{14} + x^{13} + x^{10} + x^8 + x^7 + x^6 + x^5 + x^3 + x^2 + x + 1$ |

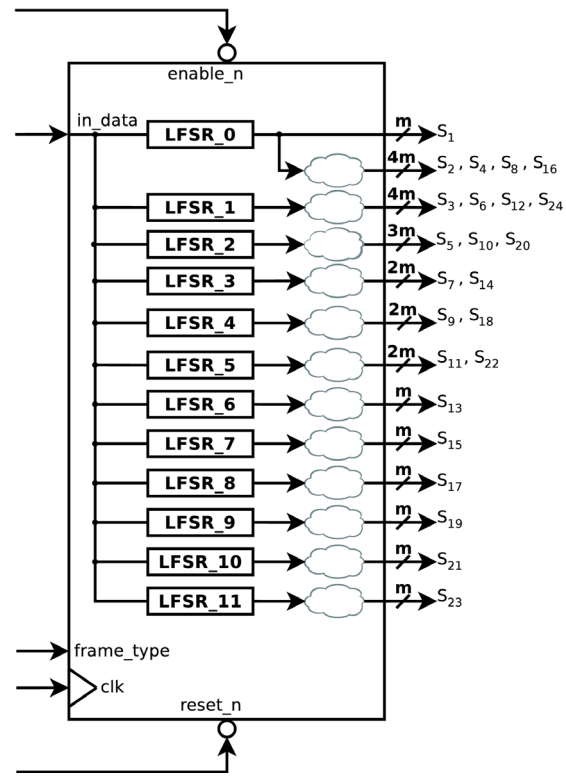


**Figure 5.** Internal architecture of the syndrome calculator

**Algorithm 1.** Combinational logic generator for the syndromes calculator.

**Input:** $i$: Index of the Galois field element $\alpha^i$ that will be substituted.
GF: GF($2^m$) type Galois Field.
$m$: Word length of each α element from GF (in bits)
**Output:** $B$: $m$ x $m$ matrix that indicates the indexes of the bits from $b_i(x)$ that have to be XORed in order to obtain $S_i$.

```
Let α^k be an element from GF
Initialize the B matrix with 0
B[1][m] ß 1
for l ß 1 to m – 1 do
    k ß (i x l) mode (2^m – 1)
    B[ l + 1 ] [ ] ß α^k
end for
return B
```

### *C. The Key Equation Solver (KES)*

This module receives the calculated syndromes and based on them, obtains a polynomial known as the Error Locator Polynomial (). The most common algorithms found in the literature for executing this stage of the BCH decoding process are the Sugiyama/ Euclidean [10], [11] and the Berlekamp-Massey (BM) algorithm [5], however, variations of the BM algorithm are more commonly used [11], [12]. These variations reduce latency by decreasing the amount of iterations for binary codes such as the BCH and also eliminate the Galois Field (GF) inversions done in each iteration of the original algorithm, thus reducing hardware complexity [13]. We implemented the Simplified inverse-free Berlekamp-Massey (SiBM), introduced in [14] and corrected by [12], which considers those two optimizations. The SiBM Algorithm is shown in Fig. 6.

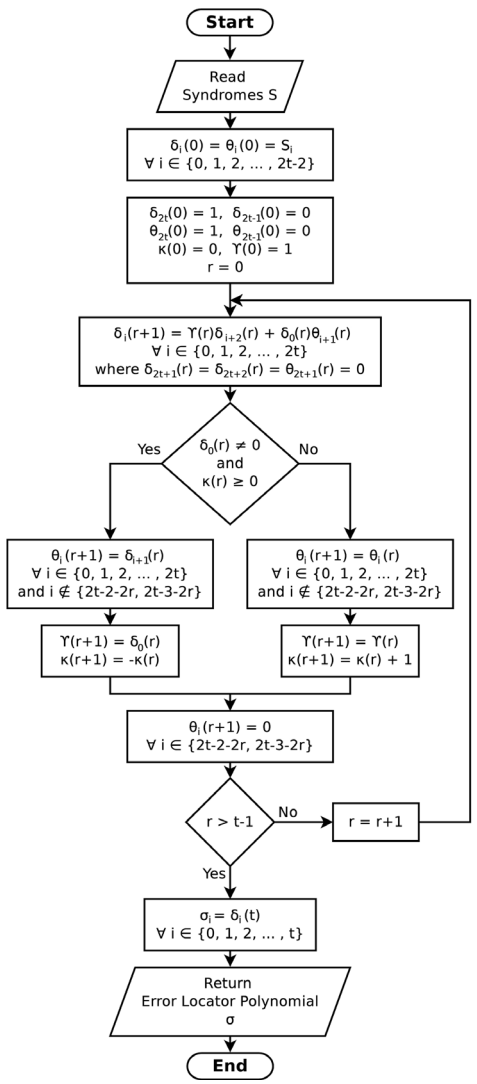


**Figure 6.** SiBM Algorithm

In the implemented circuit, the latency of this stage, in clock cycles, is equal to the error correction capacity of the decoding configuration, as established in [1] (i.e. 8, 10 or 12 clock cycles).

### *D. The Polynomial Roots Finder (PRF)*

This module uses σ to indicate which of the bits of the frame have been swapped. The algorithm implemented for this process is the Chien Search [15]. This algorithm considers σ as an algebraic expression and evaluates it by substituting each of the elements of a GF, chosen according to the size of the frame. If the evaluation of the expression is equal to zero, the GF element is considered a root of the polynomial and the position of its inverse element within the GF, the position of the error in the frame.

The DVB-S2 standard [1] considers 21 different BBFRAMES sizes classified in two groups, normal and short BBFRAMES. As established by the standard, the GF(14) and GF(16) are used by the Polynomial Roots Finder module. Since the size of each frame is smaller than its corresponding GF size, BCH codes used in DVB-S2 are considered to be shortened BCH codes. Thus, the variation of the Chien Search presented in [16], and illustrated by Fig. 7 was implemented, where $\beta = 2^m - N_{bch} - 1$, and $m = 16$ or $m = 14$ according to the frame type.

Moreover, the latency of this process can also be discriminated since it is overlapped with the latency of the transmission process.

### *E. The Memory Buffer (MB)*

This module stores a complete BBFRAME. Data is received serially at the same time as the Syndrome Calculator, but only the BBFRAME bits are stored (no BCHFEC bits). The frame is stored while the KES calculates the Error Locator Polynomial. Afterwards, the bits are fetched in a FIFO manner and XORed with the output of the PRF, in this way, the bits contained in the positions marked as erroneous are swapped back to their correct value.

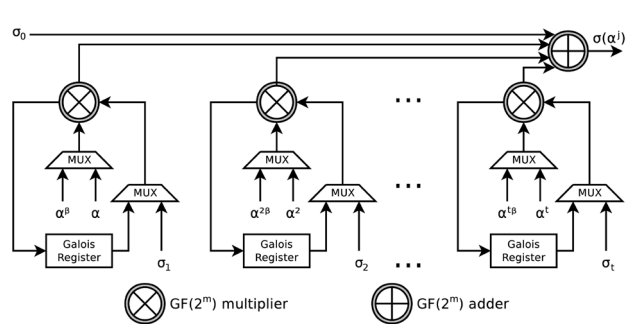


**Figure 7.** Shortened Chien search

## III. IMPLEMENTATION RESULTS

After the architecture detailed in Section II was defined, a golden model was implemented using the GNU Octave high-level interpreted language [17]. Subsequently, the BCH decoder was implemented using the VHDL hardware description language and the results of its simulations compared to those of the golden model. Afterwards, the design was prototyped in FPGA, as well as logically and physically synthesized using Encounter RTL Compiler [18], the 65nm HVT library from Global Foundries and a voltage of 1.08 v.

Although many papers related to BCH codes can be found in the literature, few of them consider long codes such as the ones used in DVB-S2. Additionally, due to the use of different technology sizes and the lack of complete synthesis information in those few papers, no fair comparisons could be made with our approach. In order to serve as a comparison base for future implementations, we present results of FPGA prototyping, logical and physical synthesis in subsections A, B, and C.

### *A. FPGA Prototype Results*

A prototype of the BCH Decoder was implemented on the Stratix IV GX FPGA development kit from Altera [19] [20]. This prototype reported a maximum frequency of 100.93MHz and a resource usage as detailed in Table III.

### *B. Logical Synthesis Results*

The amount of required cells and area occupation of the entire BCH decoder, as well as of its main modules, for a frequency of 62:5 MHz, are listed in Table IV. In the same way, leakage, switching and total power are listed in Table V.

Even though, our project requirements establish a working frequency of 62.5 MHz, we also synthesized our design for multiples of this frequency (i.e. 2x, 3x, and 4x). With this, we found out that timing violations occurred for a frequency of 250 MHz, showing a Worst Negative time Slack (WNS) of –283 ps. Results regarding the amount of cells required and area for each of the frequencies are listed in Table VI, likewise, power consumption results are listed in Table VII.

### *H. Physical synthesis results*

Values obtained at the physical synthesis stage for Internal, leakage, switching, and total power of the entire BCH Decoder die and of its main modules are presented in Table VIII for a frequency of 65.50 MHz. Power estimations for other frequencies are listed in Table IX.

Table X shows the amount of gates used for each of the four frequencies, as well as the area occupied by

**Table III.** FPGA prototyping results for Altera's Stratix IV.

| Module | Combinational ALUTs | Registers | Memory bits |
|---|---|---|---|
| SC | 868 | 192 | 0 |
| KES | 8,044 | 767 | 0 |
| PRF | 2,238 | 209 | 0 |
| MB | 87 | 37 | 58,192 |
| Total | 11,368 | 1,250 | 58,192 |

**Table IV.** Logical synthesis – 62.5 MHz utilization results.

| Module | Cells | Area ($\mu m^2$) |
|---|---|---|
| SC | 3,007 | 8,404 |
| KES | 27,265 | 76,298 |
| PRF | 7,839 | 21,715 |
| MB | 251 | 114,217 |
| Total | 38,753 | 221,660 |

**Table V.** Logical synthesis – 62.5 MHz power results.

| Module | Leakage (μW) | Switching (μW) | Total (μW) |
|---|---|---|---|
| SC | 3.05 | 1,253.78 | 1,256.83 |
| KES | 33.39 | 13,260.69 | 13,294.07 |
| PRF | 8.75 | 3,645.38 | 3,654.13 |
| MB | 32.26 | 49.11 | 81.37 |
| Total | 77.77 | 18,362.09 | 18,439.86 |

**Table VI.** Logical synthesis utilization results for other frequencies.

| Frequency (MHz) | WNS | Cells | Area ($\mu m^2$) |
|---|---|---|---|
| 62.50 | 10,223 | 38,753 | 221,660.25 |
| 125.00 | 2,275 | 38,751 | 221,706.33 |
| 187.50 | 2 | 38,766 | 221,755.61 |
| 250.00 | –283 | 40,105 | 225,342.49 |

**Table VII.** Logical synthesis power results for other frequencies.

| Frequency (MHz) | Internal (μW) | Leakage (μW) | Switching (μW) | Total (μW) |
|---|---|---|---|---|
| 62.50 | 12,786.76 | 77.77 | 18,362.09 | 18,439.86 |
| 125.00 | 21,162.43 | 77.62 | 30,628.99 | 30,706.61 |
| 187.50 | 28,438.95 | 77.54 | 41,279.20 | 41,356.73 |
| 250.00 | 34,932.68 | 80.56 | 51,341.57 | 51,422.13 |

**Table VIII.** Physical synthesis – 62.5 MHz power results.

| Module | Internal (μW) | Leakage (μW) | Switching (μW) | Total (μW) |
|---|---|---|---|---|
| SC | 350.99 | 2.99 | 770.08 | 1,124.06 |
| KES | 2,287.52 | 32.67 | 4,084.13 | 6,404.34 |
| PRF | 1,067.28 | 8.59 | 1,352.32 | 2,428.19 |
| MB | 23.17 | 32.56 | 7.46 | 62.89 |
| Total | 3,763.45 | 76.82 | 6,240.87 | 10,081.14 |

**Table IX.** Physical synthesis power results for other frequencies.

| Frequency (MHz) | Internal (μW) | Leakage (μW) | Switching (μW) | Total (μW) |
|---|---|---|---|---|
| 62.50 | 3,763.45 | 76.82 | 6,240.87 | 10,081.14 |
| 125.00 | 7,475.36 | 76.81 | 12,276.03 | 19,828.21 |
| 187.50 | 11,722.30 | 76.51 | 20,270.39 | 32,069.20 |
| 250.00 | 15,321.32 | 79.80 | 26,991.17 | 42,392.29 |

**Table X.** Physical synthesis utilization results for other frequencies.

| Frequency (MHz) | Gates | Area ($\mu m^2$) | Density (%) |
|---|---|---|---|
| 62.50 | 112,725 | 221,660.69 | 65.89 |
| 125.00 | 112,773 | 221,707.78 | 65.90 |
| 187.50 | 112,824 | 221,754.88 | 65.92 |
| 250.00 | 116,561 | 225,340.90 | 66.99 |

the entire design (including the memory block) and the density obtained in the back-end on a die of 580$\mu$m x 580$\mu$m (336400$\mu$m$^2$).

A back-end amoeba view of the BCH decoder can be seen in Fig. 8, where each of the main modules from Fig. 3 has been labeled. Additionally, a placement view can be seen in Fig. 9.

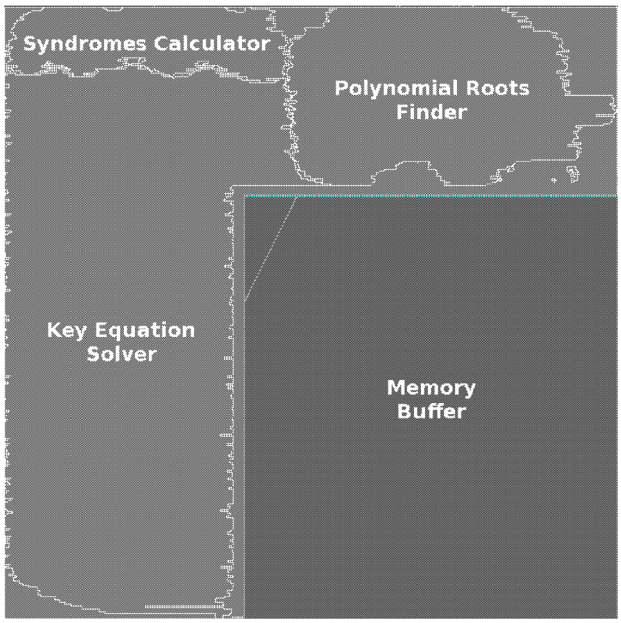


**Figure 8.** Back-end amoeba view

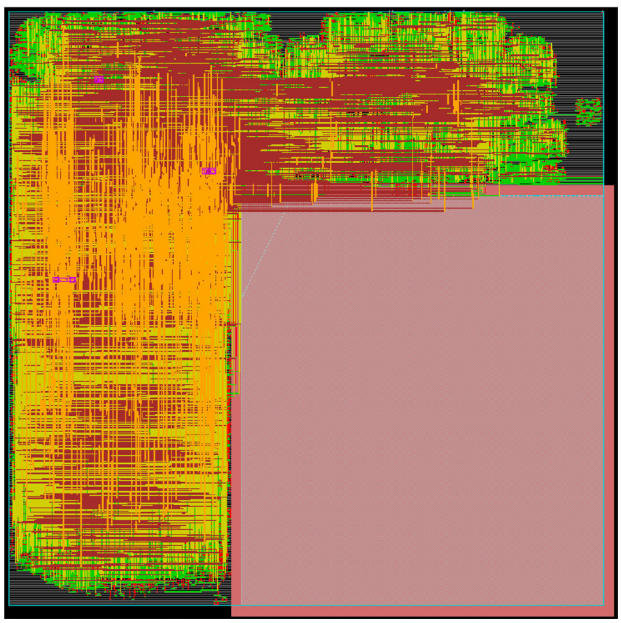

**Figure 9.** Back-end placement view

## IV. CONCLUSION

This paper presented the development of a Synthesizable BCH decoder for DVB-S2 satellite communications. The circuit was implemented in FPGA using independent platform VHDL code. Additionally, a preliminary logical synthesis for ASIC implementation was also realized, using the

Encounter RTL Compiler. Simulations were performed comparing the implementation developed in VHDL with a golden model developed using a high-level interpreted language.

Moreover, logical synthesis and Back-end results showed that the BCH Decoder circuit can be synthesized, for a frequency of 62.50 MHz, into a die size of 336400$\mu$m$^2$ with 65.89% of density with a total power of 10.08 mW using global foundry 65nm technology. Synthesis results for other frequencies were also presented.

Future work includes the study of parallel structures that can improve the latency and throughput of the proposed BCH Decoder in order to test it in other communication systems with faster error correction requirements. Moreover, area optimization will also be considered.

Additionally, future work will also consider the implementation of a variant of the proposed BCH decoder compatible with the DVB-S2X extension of the DVB-S2 standard.

## ACKNOWLEDGEMENTS

Authors would like to thank the Brazilian Ministry of Science, Technology and Innovation (MCTI), CNPq and the IC-Brazil Program, that funded this project through the grant 550467/2011-4.

## REFERENCES

[1] ETSI, “Digital Video Broadcasting (DVB); Second generation framing structure, channel coding and modulation systems from Broadcasting, Interactive Services, News Gathering and other broadband satellite applications,” ETSI EN 302 307 (V1.3.1), 03/2013.

[2] C. E. Shannon, “A Mathematical Theory of Communication,” ACM SIGMOBILE Mobile Computing and Communications Review, vol. 5, no. 1, pp. 3–55, 2001.

[3] H. Jeong and J. T. Kim, “Implementation of LDPC Decoder in DVB-S2 Using Min-Sum Algorithm,” in Convergence and Hybrid Information Technology, 2008. ICHIT ’08. International Conference on, 2008.

[4] Denise C. Alves, Eduardo R. de Lima, and Jose E. Bertuzzo, “A pipelined semi-parallel LDPC Decoder architecture for DVB-S2,” in 3rd Workshop on Circuits and Systems Design (WCAS 2013), Curitiba, Brazil, September 2013.

[5] E. R. Berlekamp, “On Decoding Binary Bose-Chadhuri-Hocquenghem Codes,” Information Theory, IEEE Transactions on, vol. 11, no. 4, pp. 577–579, 1965.

[6] H. O. Burton, “Inversionless Decoding of Binary BCH Codes,” Information Theory, IEEE Transactions on, vol. 17, no. 4, 1971.

[7] B. Zhang, D. Liu, S. Wang, X. Chen, and H. Liu, “Design and Implementation of Area-Efficient DVB-S2 BCH Decoder,” in Computer Engineering and Technology (ICCET), 2010 2nd International Conference on, vol. 3. IEEE, 2010, pp. V3–179.

[8] Y.-M. Lin, J.-Y. Wu, C.-C. Lin, and H.-C. Chang, “A Long Block Length BCH Decoder for DVB-S2 Application,” in Integrated Circuits, ISIC’09. Proceedings of the 2009 12th International Symposium on. IEEE, 2009, pp. 171–174.

[9] Cesar G. Chaves, Eduardo R. de Lima, Jacqueline G. Mertes, and Jose E. Bertuzzo, “A Synthesizable Serial-in Syndrome Calculator for DVBS2 BCH Decoding,” in 3rd Workshop on Circuits and Systems Design (WCAS 2013), Curitiba, Brazil, September 2013.

[10] G. C. Clark Jr and J. B. Cain, Error-Correction Coding for Digital Communications. Springer, 1981.

[11] P. Sweeney, Error Control Coding: From Theory to Practice. Wiley New York, 2002.

[12] M. Yin, M. Xie, and B. Yi, “Optimized algorithms for binary bch codes,” in Circuits and Systems (ISCAS), 2013 IEEE International Symposium on. IEEE, 2013, pp. 1552–1555.

[13] H.-C. Chang and C. B. Shung, “New Serial Architecture for the Berlekamp-Massey Algorithm,” Communications, IEEE Transactions on, vol. 47, no. 4, pp. 481–483, 1999.

[14] W. Liu, J. Rho, and W. Sung, “Low-Power High-Throughput BCH Error Correction VLSI Design for Multi-Level Cell NAND Flash Memories,” in Signal Processing Systems Design and Implementation, 2006. SIPS’06. IEEE Workshop on. IEEE, 2006, pp. 303–308.

[15] R. T. Chien, “Cyclic Decoding Procedures for Bose-Chaudhuri-Hocquenghem Codes,” Information Theory, IEEE Transactions on, vol. 10, no. 4, pp. 357–363, 1964.

[16] M. Gomes, G. Falc˜ao, V. Silva, V. Ferreira, A. Sengo, L. Silva, N. Marques, and M. Falc˜ao, “Scalable and parallel codec architectures for the dvb-s2 fec system,” in Circuits and Systems, 2008. APCCAS 2008. IEEE Asia Pacific Conference on. IEEE, 2008, pp. 1506–1509.

[17] John W. Eaton, “About GNU Octave,” http://www.gnu.org/software/octave/ [Accessed 14th Jun 2015].

[18] Cadence Design Systems, “Encounter RTL Compiler Datasheet,” 2012, http://www.cadence.com/rl/Resources/datasheets/encounter_ rtlcompiler.pdf [Accessed 14th Jun 2015].

[19] Altera Corporation, “Stratix IV GX FPGA Development Kit User Guide,” August 2010, http://www.altera.com/literature/ug/ug_sivgx_ fpga_dev_kit.pdf [Accessed 14th Jun 2015].

[20] Altera Corp., “Stratix IV GX FPGA Development Board Reference Manual,” August 2012, http://www.altera.com/literature/manual/rm_ sivgx_fpga_dev_board.pdf [Accessed 14th Jun 2015].