ISSN: 2277-3754
ISO 9001:2008 Certified
International Journal of Engineering and Innovative Technology (IJEIT)
Volume 5, Issue 2, August 2015

Design of the Error Detection and Correction System for the Advanced Digital Trans-receiver

Nitin S. Sonar[1], R.R. Mudholkar[2]
[1] Faculty, Engineering Department, Ibra College of Technology, Ministry of Manpower, Sultanate of Oman
[2] Professor, Department of Electronics, Shivaji University, Kolhapur, India.

ABSTRACT: Reed Solomon codes are efficient and non-binary error correcting codes. Generally the encoder is implemented using Fibonacci Linear Feedback Shift Register (LFSR) but here in this design of an encoder Galois LFSR is used to make the system parallel. Thus because of the use of Galois LFSR the propagation times are reduced. This increases the speed of execution. Also the encoder is designed in parallel to increase the overall execution speed. Galois field arithmetic is used for encoding and decoding of Reed – Solomon codes. Galois field multipliers are used for encoding the information block. The encoder attaches parity symbols to the data using a predetermined algorithm before transmission. At the decoder, the syndrome of the received codeword is calculated. VHDL implementation creates a flexible, fast method and high degree of parallelism for implementing the Reed Solomon codes. The FPGA Spartan-3E Board provides a powerful, self-contained development platform for the new parallel design.

KEYWORDS: **Fibonacci LFSR, Galois LFSR, Reed-Solomon Codes, Syndrome Calculation, FPGA.**

I. INTRODUCTION

In real world communication, errors are introduced in messages sent from one point to another (Figure 1). Reed-Solomon is an error-correcting coding system that was devised to address the issue of correcting multiple errors—especially burst-type errors—in mass storage devices (hard disk drives, DVD, barcode tags), wireless and mobile communications units, satellite links, digital TV, digital video broadcasting (DVB), and modem technologies like xDSL ("x" referring to all the existing DSL solutions, whether ADSL, VDSL, SDSL, or HDSL).

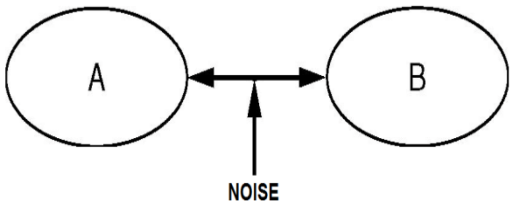


Fig 1: Two Points Exchanging Information

In order for the transmitted data to be corrected in the event that it acquires errors, it has to be encoded. The receiver uses the appended encoded bits to determine and correct the errors upon reception of the transmitted signal. The number and type of errors that are correctable depend on the specific Reed-Solomon coding scheme used. [1-4]

A Reed-Solomon code is specified as *RS(n, k)* with *s*-bit symbols, where *n* is the total number of bytes the code word contains and *k* is the number of data bytes. The number of parity bytes is equal to *n – k*, where *n* is 2 raised to the power of *s* minus one (2^{s-1}). A Reed-Solomon decoder can correct up to *t* number of bytes, where $2t = n – k$.

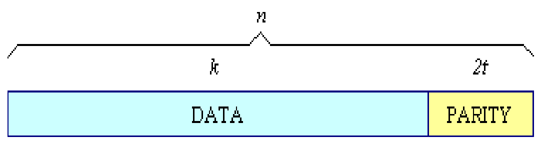


Fig 2: Reed-Solomon Code Word

Figure 2 shows a Reed-Solomon code word in which the data is left unaltered while the parity bits are suffixed to the data bits. This type of code is also known as a *systematic code*.

A well-known example of a Reed-Solomon code is *RS(255, 223)* with 8-bit symbols. For this specific Reed-Solomon code, each code word has 255 total bytes, with 223 bytes of data and 32 bytes for parity. This code has: $n = 255, k = 223, s = 8, 2t = 32, t = 16$.

This means that the decoder can automatically correct 16 symbol errors—up to 16 bytes anywhere in the code word.

A Brief Summary of Reed-Solomon Terminology:

- *Symbol_Width* is the number of bits per symbol
- *Code Word* is the block of *n* symbols
- *RS*(*n*,*k*) code: -*n* is the total number of symbols per code word -*k* is the number of information symbols per
  Code word
- *Code Rate* is equal to *k / n*
- *r = (n – k)* is the number of check symbols
- *t = (n – k) / 2* is the maximum number of symbols with errors that can be corrected.

II. FIBONACCI LFSR

In computing, a linear-feedback shift register (LFSR) is a shift register whose input bit is a linear function of its previous state. The most commonly used linear function of single bits is exclusive-or (XOR). Thus, an LFSR is most often a shift register whose input bit is driven by the XOR of some bits of the overall shift register value. The initial value of the LFSR is called the seed, and because the operation of the register is deterministic, the stream of values produced by the register is completely determined by its current (or previous) state. Likewise, because the register has a finite number of possible states, it must eventually enter a repeating cycle. However, an LFSR with a well chosen feedback function can produce a sequence of bits which appears random and which has a very long cycle. The mathematics of a cyclic redundancy check, used to provide a quick check against transmission errors, is closely related to those of an LFSR [5-10]. A 16-bit Fibonacci LFSR is shown in Figure 3. The feedback tap numbers in white correspond to a primitive polynomial in the table so the register cycles through the maximum number of 65535 states excluding the all-zeroes state.

For example, if the taps are at the 16th, 14th, 13th and 11th bits (as shown in Figure 3), the feedback polynomial is, $x^{16} + x^{14} + x^{13} + x^{11} + 1$

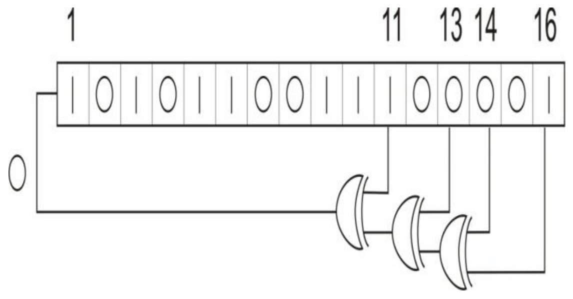


Fig. 3: Fibonacci LFSR

III. GALOIS LFSR

An LFSR in Galois configuration, which is also known as modular, internal XORs as well as one-to-many LFSR, is an alternate structure that can generate the same output stream as a conventional LFSR (but offset in time). A 16-bit Galois LFSR is shown in Figure 4. The register numbers in white correspond to the same primitive polynomial as the Fibonacci example but are counted in reverse to the shifting direction. This register also cycles through the maximal number of 65535 states excluding the all-zeroes state.

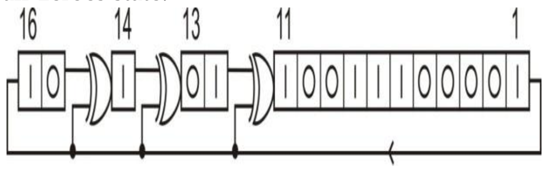


Fig 4: Galois LFSR

In the Galois configuration, when the system is clocked, bits that are not taps are shifted one position to the right unchanged. The taps, on the other hand, are XOR'd with the output bit before they are stored in the next position. The new output bit is the next input bit. Note that Galois LFSRs do not concatenate every tap to produce the new input (the XORing is done within the LFSR and no XOR gates are run in serial, therefore the propagation times are reduced to that of one XOR rather than a whole chain), thus it is possible for each tap to be computed in parallel, increasing the speed of execution [11-14].

IV. ENCODER

Reed Solomon codes are efficient and non-binary error correcting codes. Generally the encoder is implemented using Fibonacci Linear Feedback Shift Register (LFSR) but here in this design of an encoder Galois LFSR is used to make the system parallel. Thus because of the use of Galois LFSR the propagation times are reduced. This increases the speed of execution.

RS encoder should be able to perform to operations namely division and shifting. Figure 5 shows the block diagram of Reed-Solomon encoder it consist of shift registers labeled b_0 through b_{2t-1}. They are also called parity shift registers [15, 16]. Feedback data symbols are fed to the finite field multipliers labeled g_0 through g_{2t-1} are the coefficient of generator polynomial. The working of Reed Solomon encoder can be explained as follow:

1. In initial stage each parity registers will contain a value zero.
2. The message is divided into m-bit words. Each word is associated with increasing power of x forming message polynomial m(x).
3. During the first k-clock pulses switches 1 and 2 are in position B. The output will be the message symbols. During these clock pluses LFSR will compute the remainder.
4. When message symbols finished, switches are positioned to A.
5. All the parity values are shifted out of the register one at a time.

ISSN: 2277-3754
ISO 9001:2008 Certified
International Journal of Engineering and Innovative Technology (IJEIT)
Volume 5, Issue 2, August 2015

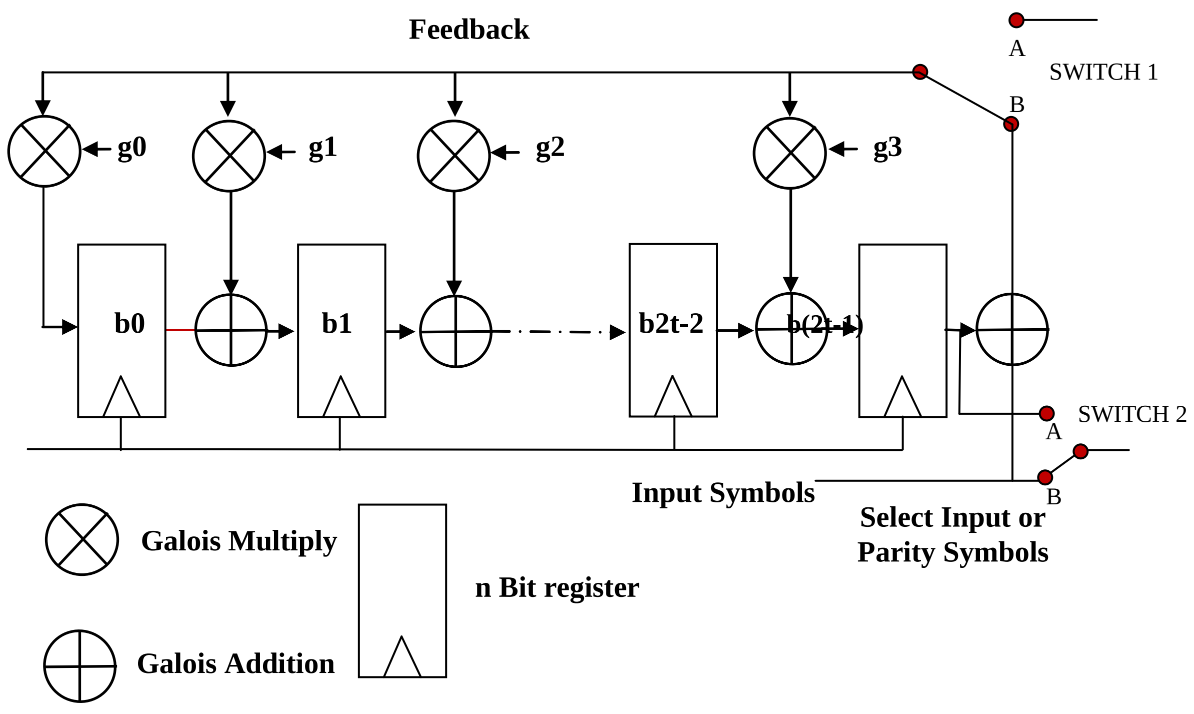


Fig 5: Architecture of RS Encoder

V. DECODER

A diagram of the Reed Solomon Decoder in detail is shown in Figure 6 below:

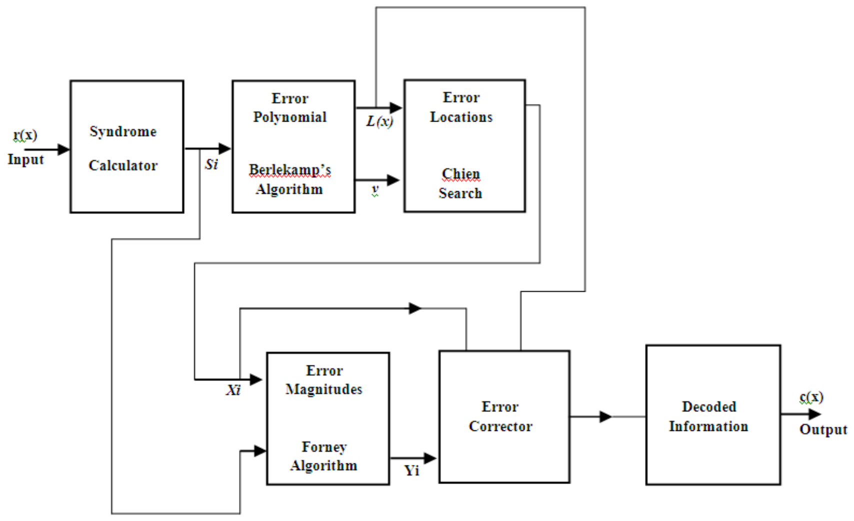


Fig 6: Block diagram of the Reed Solomon Decoder

The symbols represented in Figure 6 are:

r(x) = Received codeword

Si = Syndromes

L(x) = Error Locator Polynomial

Xi = Error Locations

Yi = Error Magnitudes

c(x) = Recovered Codeword

v = Number or errors

A Reed-Solomon codeword has 2t syndromes that depend on the errors corrected not on the transmitted codeword. The syndromes can be calculated by substituting the 2t roots of the generated polynomial g(x) into r(x), the received

ISSN: 2277-3754
ISO 9001:2008 Certified
International Journal of Engineering and Innovative Technology (IJEIT)
Volume 5, Issue 2, August 2015

codeword. The error-correcting codes become more efficient as the block size increases because the effect of noise becomes less than that for small block size. [17-21]

The Reed Solomon decoder can be implemented by using FPGA for improving the performance as well as quality of signal.

VI. IMPLEMENTATION IN FPGA

1. Syndrome calculation
2. Error Polynomial (Berlekamp-Algorithm)
3. Finding error locations using chain search algorithm.)
4. Forney Algorithm implementation
5. Error Correcting output block.

These modules can be designed using VHDL and synthesized in Xilinx ISE14.7 webpack software.

VII. SYNDROMES CALCULATOR

The Syndromes calculator block evaluates the syndromes polynomial, Si, for each incoming codeword r(x) which is considered as a series of polynomial coefficients. If Si is non-zero then the received codeword has errors and the next decoding phases are activated. If Si is zero then the received codeword is error free and no subsequent processing is needed.

The received codeword into the decoder, represented by r(x) use the original or transmitted codeword c(x) plus errors e(x). This statement is represented by the equation r(x) = c(x) + e(x). When a codeword is decoded there are three possible outcomes. First of the possibilities is when two times the amount of errors plus the erasures is less than or equal to the number of parity symbols. In the case the original transmitted code word will be recovered. Second possibilities the decoder cannot recover the code. Last possibility is the decoder can mis-decode and recover an incorrect code and have no indication of this. [22-23].

VIII. ERROR POLYNOMIAL BERLEKAMP ALGORITHM

The Error Locator Polynomial Computation block performs the evaluation of the error locator polynomial coefficients, L(x). It is based on the Berlekamp-Massey algorithm. The Berlekamp–Massey algorithm is an algorithm that will find the shortest linear feedback shift register (LFSR) for a given binary output sequence. The algorithm will also find the minimal polynomial of a linearly recurrent sequence in an arbitrary field. The field requirement means that the Berlekamp–Massey algorithm requires all non-zero elements to have a multiplicative inverse. Reeds and Sloane offer an extension to handle a ring. Elwyn Berlekamp invented an algorithm for decoding Bose–Chaudhuri–Hocquenghem (BCH) codes. James Massey recognized its application to linear feedback shift registers and simplified the algorithm. Massey termed the algorithm the LFSR Synthesis Algorithm (Berlekamp Iterative Algorithm),but it is now known as the Berlekamp–Massey algorithm [22,24-25].

ISSN: 2277-3754
ISO 9001:2008 Certified
International Journal of Engineering and Innovative Technology (IJEIT)
Volume 5, Issue 2, August 2015

IX. ERROR LOCATIONS CHIEN SEARCH

The Error Locations Computation block uses the error locator polynomial computed in the previous phase and finds the exact locations of the errors in the codeword. It is based on the Chien search time domain algorithm.In algebra, the Chien search, named after Robert T. Chien, is a fast algorithm for determining roots of polynomials defined over a finite field. The most typical use of the Chien search is in finding the roots of error-locator polynomials encountered in decoding Reed-Solomon codes and BCH codes [26, 27].

X. ERROR MAGNITUDES FORNEY ALGORITHM

The Error Magnitudes Computation block evaluates the magnitudes which are added to the received symbols in order to correct the detected errors. It is based on the Forney algorithm. In coding theory, the Forney algorithm (or Forney’s algorithm) calculates the error values at known error locations. It is used as one of the steps in decoding BCH codes and Reed–Solomon codes (a subclass of BCH codes). George David Forney, Jr. developed the algorithm [28-30].

VLSI Design Flow:

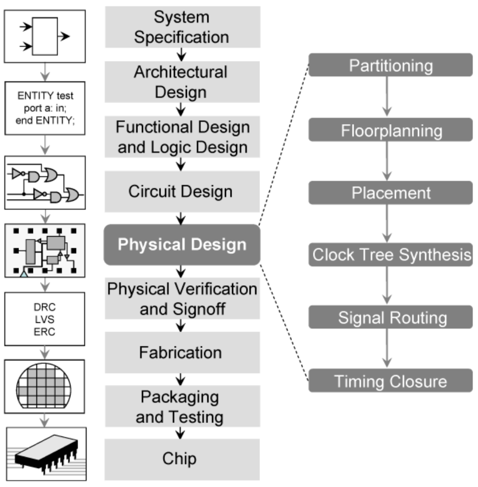


Figure 7: VLSI design flow

XI. ERROR CORRECTOR

The Error Correction block performs the actual corrections on the detected errored symbols and ge nerates the output control, status and monitoring si gnals.

XII. WHY IMPLEMENTATION OF REED SOLOMON CODE USING FPGA?

RS codecs can be implemented in a number of platforms, traditionally being Application Specific Integrated Circuits (ASICs), then shifting towards DSP. FPGAs provide the advantage of easy design reuse, faster design time especially for prototyping. In fact, programmable-hardware devices are most recommended for Reed-Solomon-codec implementation due to the abundance of registers that the hardware implementation requires.Parallel realization of equations is possible to meet time speed constraints. The RAMs available further reduce hardware resources. Equations with multipliers and power/inversion circuits may be mapped into the look-up table architecture of the FPGAs [31-35].

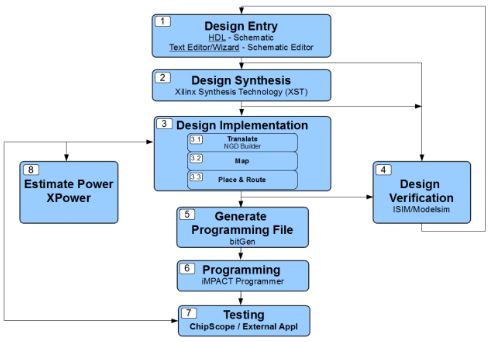


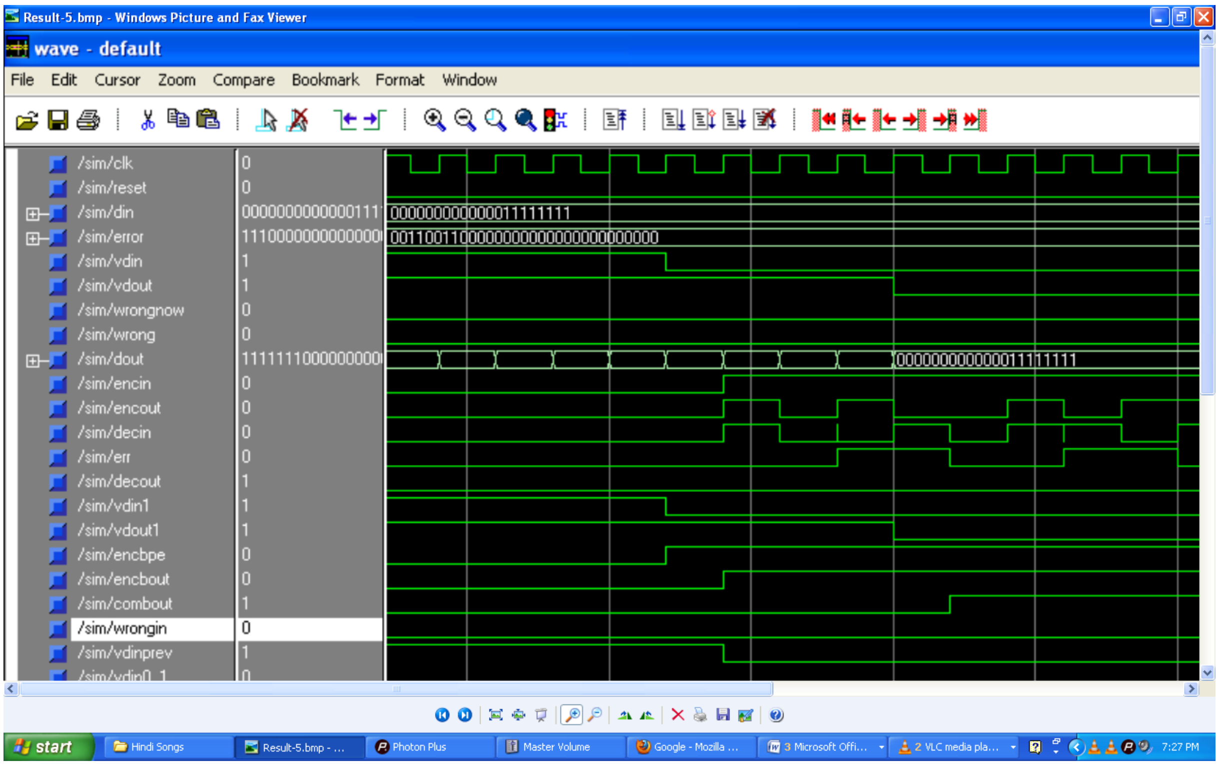
Figure 8: Circuit Design flow for FPGA

Figure 7 shows overall VLSI design flow of the system and Figure 8 shows the steps that are used to design the circuit.

XIII. RESULTS

Simulation Results

ModelSim is a tool that integrates with Xilinx ISE to provide simulation and testing. Two kinds of simulation are used for testing a design: functional simulation and timing simulation. Functional simulation is used to make sure that the logic of a design is correct. Timing simulation also takes into account the timing properties of the logic and the FPGA, so we can see how long signals take to propagate and make sure that your design will behave as expected when it is downloaded onto the FPGA.



```
/sim/clk        0
/sim/reset      0
/sim/din        000000000000000111   00000000000000011111111
/sim/error      11100000000000000    00110011000000000000000000000000
/sim/vdin       1
/sim/vdout      1
/sim/wrongnow   0
/sim/wrong      0
/sim/dout       11111110000000000    000000000000000011111111
/sim/encin      0
/sim/encout     0
/sim/decin      0
/sim/err        0
/sim/decout     1
/sim/vdin1      1
/sim/vdout1     1
/sim/encbpe     0
/sim/encbout    0
/sim/combout    1
/sim/wrongin    0
/sim/vdinprev   1
/sim/vdin0_1    0
```

Fig 9: Simulation Result-1 of a (31, 21) Reed-Solomon Corrector

Technology View

Viewing a Technology schematic opens an NGC file that can be viewed as an architecture-specific schematic. Figure 10 shows the technology schematic of RS Corrector. This schematic is generated after the optimization and technology targeting phase of the synthesis process. It shows a representation of the design in terms of logic elements optimized to the target Xilinx device or "technology"; for example, in terms of of LUTs, carry logic, I/O buffers, and other technology-specific components. Viewing this schematic allows you to see a technology-level representation of your HDL optimized for a specific Xilinx architecture, which might help you discover design issues early in the design process.

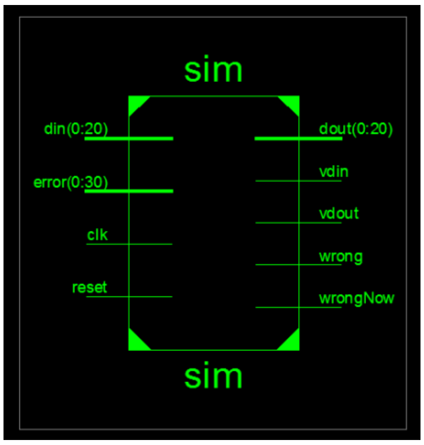


Fig 10: Technology Schematic of RS Corrector

Hardware System Results

Field Programmable Gate Arrays is used to implement this new design of the error detection and correction trans-receiver. Field Programmable Gate Arrays (FPGAs) are semiconductor devices that are based around a matrix of configurable logic blocks (CLBs) connected via programmable interconnects. FPGAs can be reprogrammed to desired application or functionality requirements after manufacturing. This feature distinguishes FPGAs from Application Specific Integrated Circuits (ASICs), which are custom manufactured for specific design tasks. Although one-time programmable (OTP) FPGAs are available, the dominant types are SRAM based which can be reprogrammed as the design evolves.

Experimental Setup with Spartan Kit

The Spartan-3 Board provides a powerful, self-contained development platform for designs targeting the

ISSN: 2277-3754
ISO 9001:2008 Certified
International Journal of Engineering and Innovative Technology (IJEIT)
Volume 5, Issue 2, August 2015

Spartan-3 FPGA from Xilinx. It features a 200K gate Spartan-3, on-board I/O devices, and 1MB fast asynchronous SRAM, making it the perfect platform to experiment with any new design, from a simple logic circuit to an embedded processor core.

The board also contains a Platform Flash JTAG-programmable ROM, so designs can easily be made non-volatile. The Spartan-3 Board is fully compatible with all versions of the Xilinx ISE tools, including the free Web Pack. The board ships with a power supply and we can add a programming cable at checkout.

The Spartan kit which is shown in Figure 11 has following important specifications,

- 18-bit multipliers, 216Kbits of block RAM, and up to 500MHz internal clock speeds
- On-board 2Mbit Platform Flash (XCF02S)
- 8 slide switches, 4 pushbuttons, 9 LEDs, and 4-digit seven-segment display
- Serial port, VGA port, and PS/2 mouse/keyboard port
- Three 40-pin expansion connectors
- Three high-current voltage regulators (3.3V, 2.5V, and 1.2V)
- Works with Digilent's JTAG3, JTAG USB, and JTAG USB Full Speed cables, as well as P4 & MultiPRO cables from Xilinx
- 1Mbyte on-board 10ns SRAM (256Kb x 32)

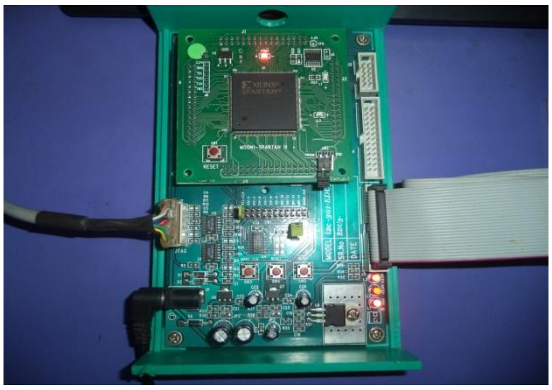


Fig 11: Spartan Kit

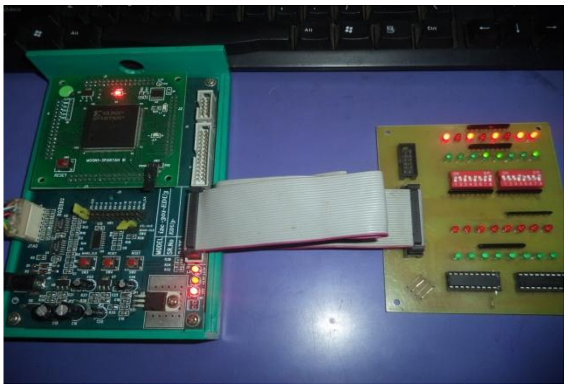
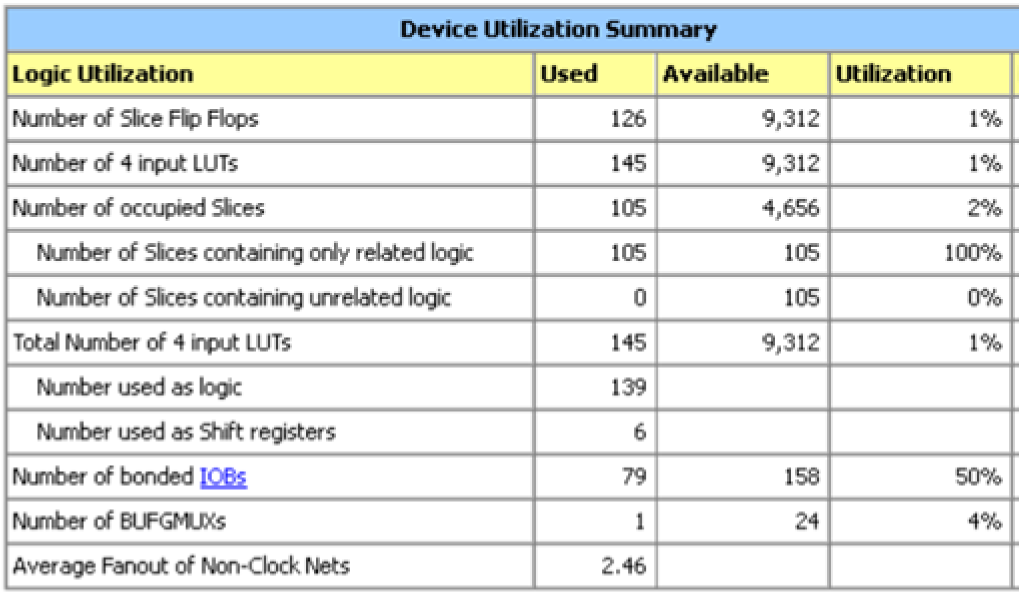


Fig 12: Experimental Setup

Table 1: Device Utilization Summary using Model-Sim



Device Utilization Summary			
Logic Utilization	**Used**	**Available**	**Utilization**
Number of Slice Flip Flops	126	9,312	1%
Number of 4 input LUTs	145	9,312	1%
Number of occupied Slices	105	4,656	2%
Number of Slices containing only related logic	105	105	100%
Number of Slices containing unrelated logic	0	105	0%
Total Number of 4 input LUTs	145	9,312	1%
Number used as logic	139		
Number used as Shift registers	6		
Number of bonded IOBs	79	158	50%
Number of BUFGMUXs	1	24	4%
Average Fanout of Non-Clock Nets	2.46		

From Table 1 we can see that the Number of slice Flip Flops utilization is only 1% i.e. only 126 Number of slice Flip Flops are used. Also we can observe that number of input LUTs and number of occupied slices utilization is only 1% and 2 % respectively. This shows that this design requires less area. Thus this new parallel architecture of Reed Solomon Error corrector reduces the occupied area of FPGA and increases the overall efficiency of the system.

RTL View

Viewing an RTL schematic opens an NGR file that can be viewed as a gate-level schematic.This schematic is generated after the HDL synthesis phase of the synthesis process. It shows a representation of the pre-optimized design in terms of generic symbols, such as adders, multipliers, counters, AND gates, and OR gates, that are independent of the targeted Xilinx device . RTL Schematic of RS Corrector is shown in Figure 13.

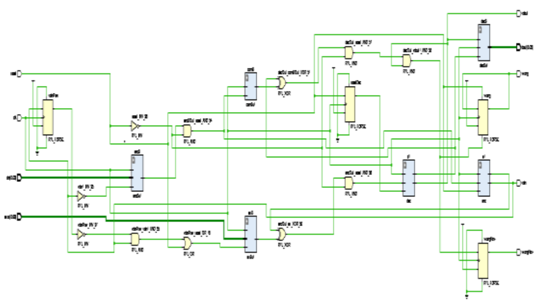


Fig 13: RTL Schematic of RS Corrector

XIV. CONCLUSION

Error control coding techniques are based on the addition of redundancy to the information message according to a prescribed rule thereby providing data a higher bit rate. This redundancy is exploited by decoder at the receiver end to decide which message bit actually transmitted. Reed-Solomon codes are an important sub–class of non binary Bose-Chaudhuri-Hocquenghem (BCH) codes.

Reed Solomon codes are efficient and non-binary error correcting codes. Generally the encoder is implemented using Fibonacci Linear Feedback Shift Register (LFSR) but here in this design of an encoder Galois LFSR is used to make the system parallel. Thus because of the use of Galois LFSR the propagation times are reduced. This increases the speed of execution. Also the encoder is designed in parallel to increase the overall execution speed. Galois field arithmetic is used for encoding and decoding of Reed – Solomon codes. Galois field multipliers are used for encoding the information block. The encoder attaches parity symbols to the data using a predetermined algorithm before transmission. At the decoder, the syndrome of the received codeword is calculated. VHDL implementation creates a flexible, fast method and high degree of parallelism for implementing the Reed Solomon codes. The new design of Reed-Solomon code presented here has following important features,

- Fully synchronous and pipelined design using a single clock.
- Detects condition when the number of errors is too high to be corrected
- Continuous, very high-speed, time-domain Reed-Solomon decoding algorithm.
- Supports different Reed-Solomon coding standards.
- Status and performance monitoring signals
- Supports continuous input data with no gaps between blocks

XV. THE FUTURE SCOPE

Coding field of communication is highly growing area in present era of research. Still, it needs a lot of improvement in the field of error correction and coding techniques. The commercial world is becoming increasingly mobile, while simultaneously demanding reliable, rapid access to sales, marketing, and accounting information. Unfortunately the mobile channel is a nasty environment in which to communicate, with deep fades an ever-present phenomenon. Reed-Solomon codes are the single best solution; there is no other error control system that can match their reliability performance in the mobile environment. The optical channel provides another set of problems altogether. Shot noise and a dispersive, noisy medium plague line-of-sight optical systems, creating noise bursts that are best handled by Reed-Solomon codes. As optical fibers see increased use in high-speed multiprocessors, we can expect to see Reed-Solomon codes used there as well.

By improving concatenation technique i.e. by concatenation of LDPC (Low Density Parity Check) as inner code and RS (Reed-Solomon) as outer coding, we can enhance the performance of FEC system. The goal of this research work is to reduce the occupied area of FPGA and increase the efficiency. The development of highly effective decoding algorithms for the implementation is another area where the significant amount of research work can be done. In more specialized, single-use applications such as the occasional deep space probe; Reed-Solomon codes will continue to be used to force communication system performance ever closer to the line drawn by Shannon. These are but a few of the applications showing that Irving Reed and Gus Solomon's codes have brought us a long way and that we can expect them to be with us for a very long time to come.

ACKNOWLEDGMENT

This work is carried out in the Digital/VLSI Lab of the Ibra College of Technology and in the VLSI Lab of the Shivaji University. Authors would like to thank all the authorities and staff of the Ibra College of Technology and Department of Electronics, Shivaji University for providing maximum facilities of the Laboratories.

REFERENCES

[1] Wicker, S. B., & Bhargava, V. K. (1999). Reed-Solomon codes and their applications. John Wiley & Sons.

[2] Koetter, R., & Vardy, A. (2003). Algebraic soft-decision decoding of Reed-Solomon codes. Information Theory, IEEE Transactions on, 49(11), 2809-2825.

[3] Guruswami, V., & Sudan, M. (1998, November). Improved decoding of Reed-Solomon and algebraic-geometric codes.

ISSN: 2277-3754
ISO 9001:2008 Certified
International Journal of Engineering and Innovative Technology (IJEIT)
Volume 5, Issue 2, August 2015

In Foundations of Computer Science, 1998. Proceedings. 39th Annual Symposium on (pp. 28-37). IEEE.

[4] McEliece, R. J. The Guruswami-Sudan Decoding Algorithm for Reed-Solomon Codes, 2003. IPN Progress Report, 42-153.

[5] Könemann, B. (1991, April). LFSR-coded test patterns for scan designs. In Proc. of European Test Conference (pp. 237-242).

[6] Krawczyk, H. (1994, January). LFSR-based hashing and authentication. In Advances in Cryptology—CRYPTO'94 (pp. 129-139). Springer Berlin Heidelberg.

[7] Krishna, C. V., Jas, A., & Touba, N. (2001). Test vector encoding using partial LFSR reseeding. In Test Conference, 2001. Proceedings. International (pp. 885-893). IEEE.

[8] Dufaza, C., & Cambon, G. (1991, April). LFSR based deterministic and pseudo-random test pattern generator structures. In Proc. European Test Conference (Vol. 199, No. 1).

[9] Goresky, M., & Klapper, A. M. (2002). Fibonacci and Galois representations of feedback-with-carry shift registers. Information Theory, IEEE Transactions on,48(11), 2826-2836.

[10] Alaus, L., Noguet, D., & Palicot, J. (2008, May). A reconfigurable linear feedback shift register operator for software defined radio terminal. In Wireless Pervasive Computing, 2008. ISWPC 2008. 3rd International Symposium on (pp. 319-323). IEEE.

[11] Joux, A., & Delaunay, P. (2006). Galois LFSR, embedded devices and side channel weaknesses. In Progress in Cryptology-INDOCRYPT 2006 (pp. 436-451). Springer Berlin Heidelberg.

[12] Dubrova, E. (2009). A transformation from the Fibonacci to the Galois NLFSRs. Information Theory, IEEE Transactions on, 55(11), 5263-5271.

[13] Mukherjee, N., Rajski, J., Mrugalski, G., Pogiel, A., & Tyszer, J. (2011). Ring generator: an ultimate linear feedback shift register. Computer, 44(6), 64-71.

[14] Joo, Y., Niu, D., Dong, X., Sun, G., Chang, N., & Xie, Y. (2010, March). Energy-and endurance-aware design of phase change memory caches. In Proceedings of the Conference on Design, Automation and Test in Europe (pp. 136-141). European Design and Automation Association.

[15] Singh, A., & Kaur, M. Study of Reed Solomon Encoder.

[16] Almeida, G. M., Bezerra, E. A., Cargnini, L. V., Fagundes, R. D. R., & Mesquita, D. G. (2007, August). A Reed-Solomon algorithm for FPGA area optimization in space applications. In Adaptive Hardware and Systems, 2007. AHS 2007. Second NASA/ESA Conference on (pp. 243-249). IEEE.

[17] Babrekar, V. J., & Sakhare, S. V. (2014). Review of FPGA Implementation of Reed-Solomon Encoder-Decoder. International Journal of Computer Applications, 87(8).

[18] Tiwari, B., & Rajesh, M. (2012). FPGA Implementation of RS Codec for Digital Video Broadcasting. V SRD-IJEECE Journal, 2(2), 68-77.

[19] Hikmat, A. (2006). Implementation of Reed Solomon Encoder/Decoder Using FPGA. Journal of Engineering and Development, 10(3).

[20] Lee, H., & Azam, A. (2003). Pipelined recursive modified Euclidean algorithm block for low-complexity, high-speed Reed–Solomon decoder. Electronics Letters, 39(19), 1371-1372.

[21] Ashenden, P. J. (2010). The designer's guide to VHDL (Vol. 3). Morgan Kaufmann.

[22] Reed, I. S., Shih, M. T., & Truong, T. K. (1991). VLSI design of inverse-free Berlekamp—Massey algorithm. IEE Proceedings E (Computers and Digital Techniques), 138(5), 295-298.

[23] DesJardins, P. A., & Mantri, R. G. (2001). U.S. Patent No. 6,219,815. Washington, DC: U.S. Patent and Trademark Office.

[24] Feng, G. L., & Tzeng, K. K. (1991). A generalization of the Berlekamp-Massey algorithm for multisequence shift-register synthesis with applications to decoding cyclic codes. Information Theory, IEEE Transactions on, 37(5), 1274-1287.

[25] Berlekamp, E. R. (1967). Factoring polynomials over finite fields. Bell System Technical Journal, 46(8), 1853-1859.

[26] Yang, H., & Gill III, J. T. (2001). U.S. Patent No. 6,192,497. Washington, DC: U.S. Patent and Trademark Office.

[27] Weng, L. J. (2002). U.S. Patent No. 6,374,383. Washington, DC: U.S. Patent and Trademark Office.

[28] Seungbeom, L. E. E., & Hanho, L. E. E. (2008). A high-speed pipelined degree-computation less modified Euclidean algorithm architecture for Reed-Solomon decoders. IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences, 91(3), 830-835.

[29] Lee, H. (2003, February). An area-efficient Euclidean algorithm block for Reed-Solomon decoder. In VLSI, 2003. Proceedings. IEEE Computer Society Annual Symposium on (pp. 209-210). IEEE.

[30] Lee, H. (2005). A high-speed low-complexity Reed-Solomon decoder for optical communications. Circuits and Systems II: Express Briefs, IEEE Transactions on, 52(8), 461-465.

[31] Elharoussi, M., Hamyani, A., & Belkasmi, M. (2013). VHDL Design and FPGA Implementation of a Parallel Reed-Solomon (15, K, D) Encoder/Decoder. International Journal of Advanced Computer Science and Applications (IJACSA), 4(1).

[32] Tiwari, B., & Mehra, R. (2012, March). Design and implementation of Reed Solomon Decoder for 802.16 network using FPGA. In Signal Processing, Computing and Control (ISPCC), 2012 IEEE International Conference on (pp. 1-5). IEEE.

[33] Paar, C., & Rosner, M. (1997, April). Comparison of arithmetic architectures for Reed-Solomon decoders in

reconfigurable hardware. In Field-Programmable Custom Computing Machines, 1997. Proceedings. The 5th Annual IEEE Symposium on (pp. 219-225). IEEE.

[34] Almeida, G. M., Bezerra, E. A., Cargnini, L. V., Fagundes, R. D. R., & Mesquita, D. G. (2007, August). A Reed-Solomon algorithm for FPGA area optimization in space applications. In Adaptive Hardware and Systems, 2007. AHS 2007. Second NASA/ESA Conference on (pp. 243-249). IEEE.

[35] Haase, A., Kretzschmar, C., Siegmund, R., Müller, D., Schneider, J., Boden, M., & Langer, M. (2002, March). Design of a Reed-Solomon Decoder using Partial Dynamic Reconfiguration of XILINX Virtex FPGAs-A Case Study. In Proceedings of DATE (pp. 151-157).

AUTHOR BIOGRAPHY

Prof (Dr.) R. R. Mudholkar received the M.Sc. degree in 1984 form Karnataka University, Dharwad, M.Phil.in 1996, Ph.D. degree in 2003 from Shivaji University, Kolhapur. He is currently working as a Professor in Electronics Department, Shivaji University, Kolhapur. His research interest includes Soft Computing, Hardware Implementation of Fuzzy Systems and Electronic Neural Network. He is guiding Ph.D. and M. Phil. Students of Electronics and Computer Science. He has 80 Research Papers published in National and International journals. He has delivered good number of invited talks at National/International level Workshops, Conferences and Seminars.

Mr. Nitin S. Sonar received M.Sc. (Electronics), PG.Dip. VLSI and M.Phil. (Electronics) degrees from Shivaji University, Bit Mapper Institute and Pune University, India respectively in 1997, 2001, and 2009 respectively. He has scored Rank First in M.Sc.(Electronics) at the University level. He has scored Grade 'A' in both P.G. diploma in VLSI, and in M.Phil. (Electronics). Since 2009 he has been with Ibra College of Technology, Sultanate of Oman, where he is currently a distinguished Lecturer in the department of Electrical and Electronics Engineering. His research addresses VLSI architecture design and implementation of data communications systems. He is pursuing Ph.D. degree in Electronics form Shivaji University, India. He is currently working on error control coders, high speed trans-receivers using VLSI technology.